# Learning Search Strategies from Human Demonstration for Robotic Assembly Tasks

Dennis Ehlers

**School of Electrical Engineering**

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 8.8.2018

**Supervisor**

Prof. Ville Kyrki
Prof. George Nikolakopoulos

**Advisors**

MSc Markku Suomalainen

MSc Jens Lundell

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

**Author** Dennis Ehlers

**Title** Learning Search Strategies from Human Demonstration for Robotic Assembly Tasks

**Degree programme** Erasmus Mundus Master in Space Science and Technology

**Major** Space Robotics and Automation **Code of major** ELEC3047

**Supervisor** Prof. Ville Kyrki Prof. George Nikolakopoulos

**Advisors** MSc Markku Suomalainen, MSc Jens Lundell

**Date** 8.8.2018 **Number of pages** 66 **Language** English

**Abstract**

Learning from Demonstration (LfD) has been used in robotics research for the last decades to solve issues pertaining to conventional programming of robots. This framework enables a robot to learn a task simply from a human demonstration. However, it is unfeasible to teach a robot all possible scenarios, which may lead to e.g. the robot getting stuck. In order to solve this, a search is necessary. However, no current work is able to provide a search approach that is both simple and general. This thesis develops and evaluates a new framework based on LfD that combines both of these aspects. A single demonstration of a human search is made and a model of it is learned. From this model a search trajectory is sampled and optimized. Based on that trajectory, a prediction of the encountered environmental forces is made. An impedance controller with feed-forward of the predicted forces is then used to evaluate the algorithm on a Peg-in-Hole task. The final results show that the framework is able to successfully learn and reproduce a search from just one single human demonstration. Ultimately some suggestions are made for further benchmarks and development.

**Keywords** Learning from Demonstration, Robotics, Robotic Assembly, Search Strategies, Learning Search, Compliant Motions

# Preface

This work was carried out at the Intelligent Robotics group at Aalto Univeristy and I would like to show my gratitude to all the amazing people there who have supported me during this thesis. As such, I would first of all very much like to thank my supervisor Ville Kyrki. With your knowledge and experience you always helped me out and put me on the right track. Thanks also to my advisors Markku Suomalainen and Jens Lundell. I've bugged you so much about all too many things and you nevertheless were always happy to help me out with your advice. I also want to thank all the other members of the Intelligent Robotics group. You have made me feel at home in the group right away and made my work there so much more enjoyable! Ultimately I'd also like to thank my friends as well as my family, especially my dear brother Sven. You have all supported me so much throughout this thesis and I wouldn't have managed this without you!

Otaniemi, 8.8.2018

Dennis Ehlers

# Contents

# Abbreviations

| | |
|---|---|
| LfD | Learning from Demonstration |
| PbD | Programming by Demonstration |
| GMM | Gaussian Mixture Model |
| HMM | Hidden Markov Model |
| BIC | Bayesian Information Criterion |
| EM | Expectation-Maximization |
| F/T | Force / Torque |
| FPI | Fitted Policy Iteration |
| POMDP | Partially Observable Markov Decision Process |
| TCP | Tool Center Point |
| TSP | Travelling Salesman Problem |
| HW | Hardware |
| SW | Software |
| KRC | KUKA Robot Controller |
| DLR | Deutsches Zentrum für Luft- und Raumfahrt — German Aerospace Center |
| DOF | Degrees of Freedom |
| KRL | KUKA Robot Language |
| FRI | Fast Research Interface |
| OROCOS | Open RObot COntrol Software |
| ROS | Robot Operating System |
| OPS | OROCOS Programming Script |

# 1 Introduction

Since the beginning of time, humans have invented and enhanced machines that improve their quality of life. In this day and age, almost every aspect of our world is deeply dependent on advanced machinery. One of the most helpful type of such machines is robots. They come in a sheer variety of different shapes and sizes, from simple vacuum cleaning robots to those that perform surgery. The most common type of robot is however that of the industrial robot manipulator, as they automate many laborious tasks. These kind of robots are often seen in the assembly halls of automobile production lines, where they follow strict routines and fixed waypoints in order to fulfill their assembly goals. The fact that these tasks are known beforehand make these robots predestined for such environments, which explains their high numbers. The static environment of these tasks allows for easy programming, since all the positions and orientations of the robots' joints were already determined. However, when the complexity of assembly tasks increases and the environments become more dynamic, it becomes unfeasible to manually program all the necessary positions, velocities and torques. In these situations it is necessary to develop new methods for programming robots. This was the focus of ongoing research for the last decades.

One of the popular methods that is researched is to let a human demonstrate a task to the robot and make the robot learn from this demonstration. This is called *Learning from Demonstration* (LfD) or *Programming by Demonstration* (PbD) [9]. In the LfD approach, a human person demonstrates a task to a robot and the robot will then learn from this demonstration what to do in any given situation [9]. This method of teaching a task to a robot is getting more and more common. However, since it is impossible to teach a robot all possible scenarios, it can for example get stuck during an assembly task or it may lose track of its current position. In these cases the need for search strategies arises.

There exist approaches towards search strategies in assembly tasks, but to the knowledge of the author there are unfortunately no methods which are both simple and general enough to be used for any environment. Often, the search motion is merely a deterministic trajectory, such as an Archimedean spiral [21] [22] [23]. These methods require the user to tailor the search to each environment by hand which is a laborious process and time consuming and is only feasible in a 2D environment. More recently, there were works published on achieving coverage of the search area by using ergodic motions [25] [26], i.e. the search intensifies in the areas with a higher amount of information. This approach however requires a complex framework to be in place.

To solve this issue, a new approach is necessary that combines a simple framework with a general field of assembly tasks. One such approach is to learn how to cover the search area from observing a human's search motions. The only similar approach uses POMDPs to model the search [20], however this is not only a complex approach,

it also requires a high amount of demonstrations (300). Not only does this cause an unfeasible amount of wear and tear on the robot, it is also a time-intensive and laborious process. Thus the question is as follows: Is it feasible to develop a simple approach that can learn search motions from human demonstration with only a single demonstration? This will be the question that this thesis aims to answer.

In order to provide this answer, first a learning framework and corresponding controllers were developed. Then, to ensure the proper choice of learning framework and controllers, the effects of their different configurations are investigated. Finally the learning framework is benchmarked in a Peg-in-Hole task.

The results of the thesis show that it is in fact possible to learn successful search motions from a single human demonstration. However, while the thesis intends to propose a framework that works for any dimensionality, the evaluation of the work is limited to a 2D environment.

The structure of this thesis is as follows. Chapter 2 provides the reader with the necessary background knowledge of the thesis' main aspects, and reviews the literature on these topics. In Chapter 3, the learning framework is explained. The setup used for the experiments is described in Chapter 4. Chapter 5 presents the outcome and investigates the effects the learning frameworks' parameters have on it. Finally, the thesis is concluded in Chapter 6 by describing its main outcome.

# 2 Background

## 2.1 Learning from Demonstration

Traditionally, robots are directly programmed for a specific task, meaning the user had to specify details, such as joint torques, velocities and positions in order to make the movements required for the task [1]. This approach, however, is a laborious one in the case of dynamic environments: tasks might differ in terms of, for example, working environment or workpiece placement. They thus require high expertise on the side of the programmer. To overcome such problems, a technique called *Learning from Demonstration* (LfD), also known as *Imitation Learning*, or *Programming by Demonstration* (PbD) [9], is used. With this method, the programmer, also called teacher or demonstrator, can demonstrate a task to the robot, which is sometimes called the student. During this demonstration the states, *e.g.* joint angles and sometimes even the actions, *e.g.* joint torques, are being recorded. This means the states and actions recorded result directly from the user's input. However, for the robot to reproduce the motions with its own actuators, the mapping from the user's input to the robot-actuator output needs to be solved. In order to achieve precisely this and resolve the mapping-issue, Learning from Demonstration is used [2].

### 2.1.1 Types of demonstrations

The first aspect to be reviewed is the demonstration part of the LfD approach, where one important question is: who is performing the demos and how is the data recorded? This is important because a direct mapping from demonstration to reproduction is not always possible, especially with a human teacher [2]. This problem of mapping between teachers and students of different morphologies is called the correspondence problem [3], and it relates to data recording:

States and actions can be recorded in two ways: 1) *Demonstration*, where all states and actions are recorded on the actual robot platform, and 2) *Imitation*, where data are recorded on a platform that is different from the one used for reproduction [2]. These categories can be split even further; however, since this thesis only deals with the Demonstration approach, the Imitation method is not be discussed further. The interested reader can however get more information about it in the survey of robot learning from demonstration by Argall *et. al* [2, Chapter 3.3].

Depending on how the demonstration is performed, it can be of the *Teleoperation*, *Shadowing* or *Kinesthetic Teaching* class [2]. In the Teleoperation approach, the demonstrator operates the robot with an input device such as a joystick. This approach assures a direct mapping between the teacher's input and the student's reproduction, since all the states the robot is in and all the actions it executes are directly recorded. However, the more complex the robot gets, the more difficult it is to control via Teleoperation. Prime examples for the use of Teleoperation in teaching include autonomous helicopters [4], UAVs [5], as well as self-driving cars [6].

In the Shadowing approach, the student mimics the teacher's motions during demonstrations. This can be done for example as shown in Figure 1, where a human teacher equipped with sensors demonstrates a pose, and the robot learner simultaneously mimics their motions. However, the teacher's motions are not recorded during the demonstration, only the learner's. Thus to be able to shadow the teacher's motions live, an additional component is necessary in the learning process. As a result it is impossible to directly record the teacher's actions using the robots internal sensors as in the Teleoperation approach. Thus, in the Shadowing approach, the recorded actions are the ones that are executed during the mimicking of the demonstrated motions [2]. This means the recording contains a non-identical mapping between the human teacher and the robot student, further complicating the method. Nevertheless this approach is used within a variety of settings; examples include a robot following an identical counterpart through a maze [7] or a humanoid robot learning motions from a human partner [8].

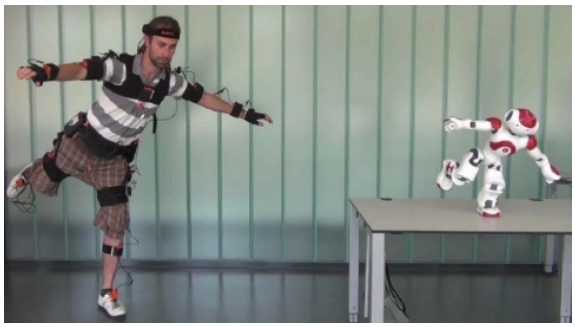


Figure 1: The teacher seen on the left has various sensors attached to them to measure their pose. In the Shadowing approach, the robot then tries to mimic the motions experienced by the teacher's sensors. (Source: https://www.cc.gatech.edu/ chernova/CS7633/slides/LfDOverview.pdf)

Finally, the third class, *Kinesthetic Teaching*, refers to the user guiding the robot through the desired motions by keeping constant physical contact with it [9]. Figure 2 shows how such a guided demonstration can look like. One of the advantages of this approach is that there is no difference between the demonstrated and the learned states and actions, as the internal sensors directly record the data from the demonstration. Additionally, unlike the Teleoperation method, no control device is used and the user can thus simply manipulate the robot by manually moving it with their hands. Therefore even users that have little to no experience with robots can act as teachers and demonstrate tasks. However, there are a series of disadvantages with this approach, such as the need to have a robot that can compensate for the gravity acting on it. In addition, the teacher usually uses more of his joints than there are on the robot, for example when they move a single robotic arm with both their hands. This is also a disadvantage vice-versa: it is difficult to teach motions to robots that have the same number of limbs, or even more, than the demonstrator [9]. However, the joint configuration of the robot used in this thesis, a KUKA LWR4+

(see Section 4.1), makes it possible to comfortably guide it with two human hands. Thus, in addition to the above mentioned simplicity of mapping input to output, Kinesthetic Teaching was chosen as the used demonstration technique. Additionally, it is shown that this approach works well for the used robot [15, 16] and is perceived by users as easier to use in comparison to Teleoperation [28] [30].

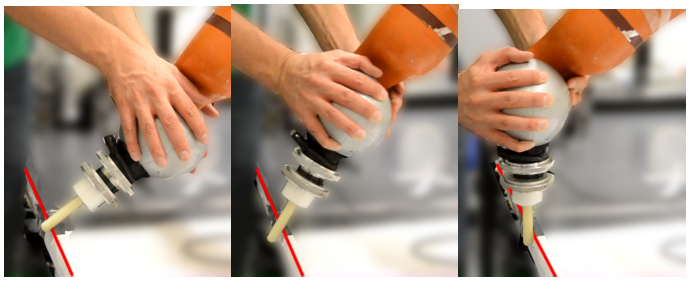


Figure 2: In the Kinesthetic teaching approach, the user physically guides the robot through the desired motions. (Source: [16])

### 2.1.2 Methods of learning a policy

In the Learning from Demonstration technique, the demonstration is, as the name suggests, only one part of the whole approach. Since it is not feasible to demonstrate all possible states the robot can be in, it is necessary to extrapolate the actions the robot executes. By using the demonstrated state-action pairs, a model of the world can be learned. The robot then needs to learn what kind of action to do in a certain state. This mapping from state to action is called a *policy*. The policy is learned from the recorded state-action pairs and enables the robot to reproduce the trajectory [2]. To be able to learn a policy from the demonstrations, the recorded data can be modeled. To build such a model, one can refer to different approaches.

On the lowest level a model can be represented as a Gaussian Mixture Model (GMM), such as in the method proposed by Calinon *et. al.* [14]. The paper describes possible modelling solutions of the LfD approach, and presents modelling with GMMs as one such solution. In a GMM a dataset is modeled by a combination of several different Gaussian distributions. A more detailed explanation of GMMs can be found in Chapter 3.2. To estimate the number of Gaussians in the GMM, they use the Bayesian Information Criterion. To train the GMM, they use an expectation-maximization algorithm in order to estimate the most likely, i.e. the best-fitting, GMM parameters. The experimental results showed that by using a GMM a robot could successfully reproduce a peg-in-hole task. Based on this, and since the experimental task in this thesis also targets peg-in-hole tasks, this approach was chosen to learning a policy for this thesis.

Additionally, these low level solutions can be used in order to build an even more complex model. For example, the paper by Hagos *et. al.* proposes a non-homogeneous

Hidden Markov Model (HMM) to model an assembly task [29]: They learn how to reproduce a human-demonstrated assembly task by segmenting human motions into several compliant phases and using an HMM to model these different phases. To learn how many phases the demonstration should be segmented, they use the Bayesian Information Criterion (BIC), which is explained more in-depth in Section 3.2. Each phase, whose state dynamics are approximated by a linear Gaussian model, is then modeled as one hidden state of the Hidden Markov Model. They then use an expectation-maximization (EM) algorithm to learn the necessary parameters from multiple human demonstrations. While this approach works well for their application, the long runtime of the EM algorithm, and the complexity of the HMM are unnecessarily complicated for this thesis, which is why a simpler method of modelling was chosen.

## 2.2 Compliant Motions

The prospect of automating high accuracy assembly tasks is still popular, with much emphasis on dealing with incomplete information about the state of the robot or the workpiece. This missing information can stem from, for example, irregularities in the placement of the workpieces or vision systems that fail to produce a precise measurement of the relative pose between robot and workpiece [17, 18, 19]. One approach that deals with these inaccuracies is *compliant motions*, that is motions caused by the interaction of the forces executed by, *e.g.*, a human or a robot, and the counterforces from the environment. For example, when trying to insert a key into a keyhole, a human naturally applies forces and torques in such a manner that their movements are compliant to the environment. This is possible because of a human's inherent sensing ability of forces generated by the environment. An simplified example of this can be seen in Figure 3, where the compliant motion – caused by the interaction between a rightwards directed force on the workpiece and the resulting environmental forces – guides the workpiece on the left into the hole.

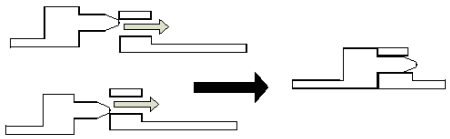


Figure 3: This figure shows the use of compliant motions to complete an assembly task. (Source: [15])

One of the main works that this thesis builds upon are the papers on the learning of compliant motions by Suomalainen and Kyrki [15] [16]. They propose a method to learn compliant motions from human demonstrations, taking advantage of the environment to perform positionally and orientationally compliant motions. Since the experiments in this thesis do not include orientational control, the main focus is be on the positional-only compliance demonstrated in [15]. However, the method proposed in this thesis is nevertheless devised to work in 6-D as well. In order to

achieve their goal, Suomalainen and Kyrki start with making demonstrations and recording the interaction forces, i.e. the forces the environment exerts on the robot, with a Force- and Torque sensor (F/T sensor) mounted between the tool and the area of the robot where the user places their hands on to guide the robot. Then they define the set of desired directions $\hat{\boldsymbol{v}}_{\boldsymbol{d}}$ as the directions between the negative of the force measured by the F/T sensor, $-\hat{\boldsymbol{F}}_{\boldsymbol{m}}$, and the direction of motion $\hat{\boldsymbol{v}}_{\boldsymbol{a}}$. A force in any of those desired directions eventually leads towards the corresponding desired position, assuming a static environment. After learning the demonstrated desired direction $\boldsymbol{v}_{\boldsymbol{d}}^{*}$ out of the set of $\hat{\boldsymbol{v}}_{\boldsymbol{d}}$ that leads towards the desired position, they also identify the fully compliant axes. Fully compliant axes possess no stiffness and the robot does not move if commanded in the direction of those axes. The only motion along the compliant axes is induced by the environment. For reproducing the learned motions, they use an impedance controller defined as follows:

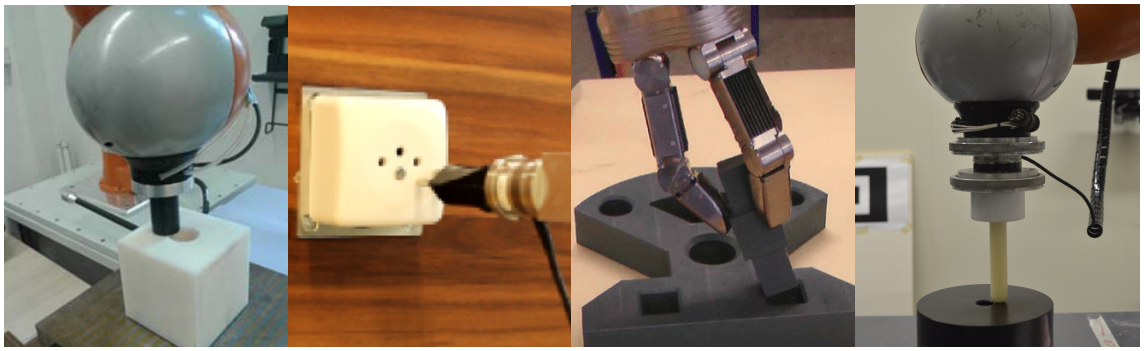
$$\boldsymbol{F} = \boldsymbol{K}(\boldsymbol{x}^{*} - \boldsymbol{x}) + \boldsymbol{D}\boldsymbol{v} + \boldsymbol{f}_{\boldsymbol{dyn}} \ , \tag{1}$$

where $\boldsymbol{K}$ is the stiffness that defines how much force corresponds to a certain displacement, which is set according to the compliant axes and desired direction learned earlier, $\boldsymbol{x}^{*}$ is the desired position, $\boldsymbol{x}$ the current position, $\boldsymbol{D}$ the damping coefficient, $\boldsymbol{v}$ the current velocity and $\boldsymbol{f}_{\boldsymbol{dyn}}$ are the feed-forward dynamics of the robot [15]. Using this method, they successfully learn the motions required to reproduce individual phases of assembly tasks from one or more demonstrations. However, this approach only works if there is a desired direction and a guiding environment. In cases where no desired direction can be discerned, or the environment is not guiding towards the goal, another approach is needed. One type of scenario where this is the case is Peg-in-Hole assembly tasks, which are presented in the following chapter.

## 2.3 The peg-in-hole task

The peg-in-hole class of assemblies is a popular task in the robotics community [33, 37, 32, 24] and relies heavily on compliant motions. Many different solutions were proposed: from using Learning from Demonstration [16] [23], to approaches supported by vision systems [34] [35], to reinforcement learning algorithms that use CAD models as an information source [36] [27]. While the specific setup of the peg-in-hole task can vary with each application (see Figure 4), the general approach is roughly the same, namely the insertion of one object into a hole of similar shape. Insertion is facilitated by using compliant motions, as they allow the peg to be inserted from different angles and positions. The setup used in this thesis is the same as in [16] and is seen in Figure 4 (d). The spherical end of the peg helps insofar that, assuming a steady downwards directed force on the peg, only the very tip of the peg needs to reach the hole to be inserted, as opposed to the peg completely covering the hole with its complete circumference. Thus, the effective diameter of the goal area becomes bigger. If the peg is within that area and under a constant level of vertical force, it slides down into the hole and the task is completed.

Although before the peg can be inserted, the hole first needs to be located. Generally, any assembly task where the goal is unknown requires compliant searching for the goal. There are however many different methods of searching, which is why the following chapter will review the currently used search strategies.



(a) (b) (c) (d)

Figure 4: This figure shows different Peg-in-Hole tasks. The common goal across all of them is to insert one object into the hole of similar shape. Figure (d) shows the implementation of the Peg-in-Hole task used in this thesis. (Sources: (a): [21], adapted; (b): [19], adapted; (c): [51], adapted)

## 2.4 Search Strategies

There exist many different methods for locating goals of an unknown position. Some focus on a deterministic approach [21] [22] [23], others are of a stochastic nature [24], and yet others propose an ergodic approach [25] [26]. There are even methods based on CAD models [27]. In this section, these strategies are reviewed in-detail.

The work by de Chambrier *et. al.* [20] focuses on solving a peg-in-hole task in a visionless environment, by learning a belief-space value function from human demonstration via a fitted policy iteration (FPI) framework. By making use of FPI, they can model the problem as a large and continuous partially observable Markov decision process (POMDP). More specifically, the goal of the task is to locate a power outlet socket on a wall with a corresponding plug and inserting the plug into the socket. To achieve this, 10 different test subjects made 15 demonstrations each for two geometrically different types of sockets, resulting in 300 overall demonstrations. With a reported average duration of 50 minutes per test subject, it stands to reason that it took a little over 8 hours to collect the data from all necessary demonstrations. This high amount of time to demonstrate is inconvenient and expensive, which is why this thesis investigates how feasible the use of just one single demonstration is for a peg-in-hole task. Furthermore, while the demonstrators did not know the relative location of the socket with respect to their position, the absolute location of the socket on the wall was in fact known. This leads to the behaviour shown in their experiments, where, if the socket was positioned on the upper right corner

of a wall, the robot would immediately search in this corner to locate the socket, as can be seen in Figure 5. However, there are many possibilities and tasks where the goal location is not known and there are no environmental guides present. In these situations the method in [20] would fail, as no information is present about the relative location of the target. In contrast, the method presented in this thesis makes no assumptions about the location of the goal and is thus suited for situations with no guiding environment.

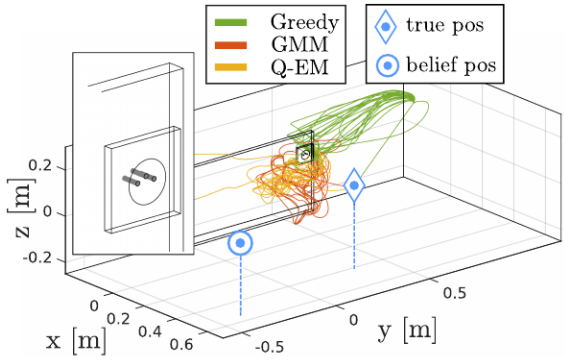



Figure 5: This figure shows the search strategies implemented by [20]. (Source: [20])

Another approach that deals with an unknown goal position is presented by Jasim *et. al.* [21]. By constructing a search trajectory in the form of an Archimedean spiral as seen in Figure 6, they can cover all possible goal locations. To be more precise, they describe a peg-in-hole task, where the peg starts off in mid-air and descends downwards onto a platform with a hole. Should the peg upon contact miss the hole, a pre-calculated search-trajectory in spiral shape is be followed, until a contact-state based detection algorithm recognized that the peg has reached the hole [21]. This process is explained in more detail in [22, Chapter 4.2.1]. The advantages mentioned in favour of using the spiral trajectory over other paths are its simplicity, and the fact that the path, if programmed correctly, eventually reaches the hole [22, Page 54]. They also mention that the spiral path locates the hole faster than other paths [22, Page 55]. It is worth pointing out that the time until the hole is located depends on the choice of parameters, specifically on the spanning distance of the spiral. If the hole has to be reliably located, the fastest time possible is achieved when the spanning distance is precisely twice the clearance of the assembled peg-in-hole, i.e. the distance between the inner wall of the hole and the outer surface of the peg [22, Page 56]. However, even though this approach is described as the fastest[22, Page 55], it is deterministic and potentially spends a lot of time in areas where a human might not search extensively. As such, it stands to reason that an algorithm that learns the search space and -motions from a human demonstration can lead to, at least on average, faster search motions. In addition, using a deterministic search path means that for each use case the path has to be individually programmed for the corresponding environment. Even more so, the environment has to be known completely in order to build a path that extensively covers it, an assumption that

does not always hold in cases such as an obstacle appearing during the search. The method proposed in this thesis does not use a deterministic path, nor does it require any knowledge about the environment. In addition, the spiral search path is only feasible to implement in a 2D environment, whereas the method presented in this thesis should work with any number of dimensions.

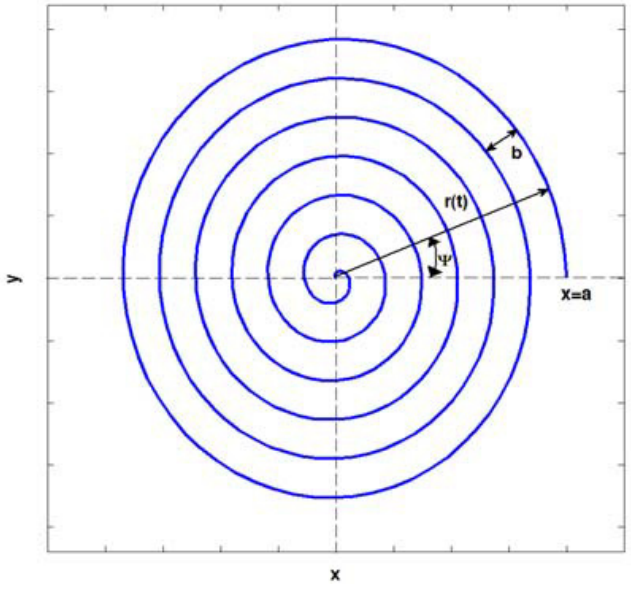



Figure 6: With the correct parameters, a spiral search path is guaranteed to locate the hole. (Source: [21])

An example of a non-deterministic search strategy is the approach by Miller and Murphey [26]. They propose a trajectory chosen on the basis of ergodicity, which is basically a measure of a sample's representativeness of the overall underlying distribution. This means that an ergodic search trajectory, with respect to a human demonstration, is sampled in such a way that it covers the search space in a similar, but not equal way, as the human demonstration. Based on this measure, Miller and Murphey present a method that creates a trajectory that covers an exploration region in a ergodically optimal way by making use of a probability density function that contains some measure of information density over said region [26]. Figure 7 shows an example of an ergodically sampled trajectory. While this is a good method to learn the search motions from a human, it is however complicated in the way the model is created, which is why this thesis proposes a simpler method of modelling the search-space and sampling from it in a human-like way.

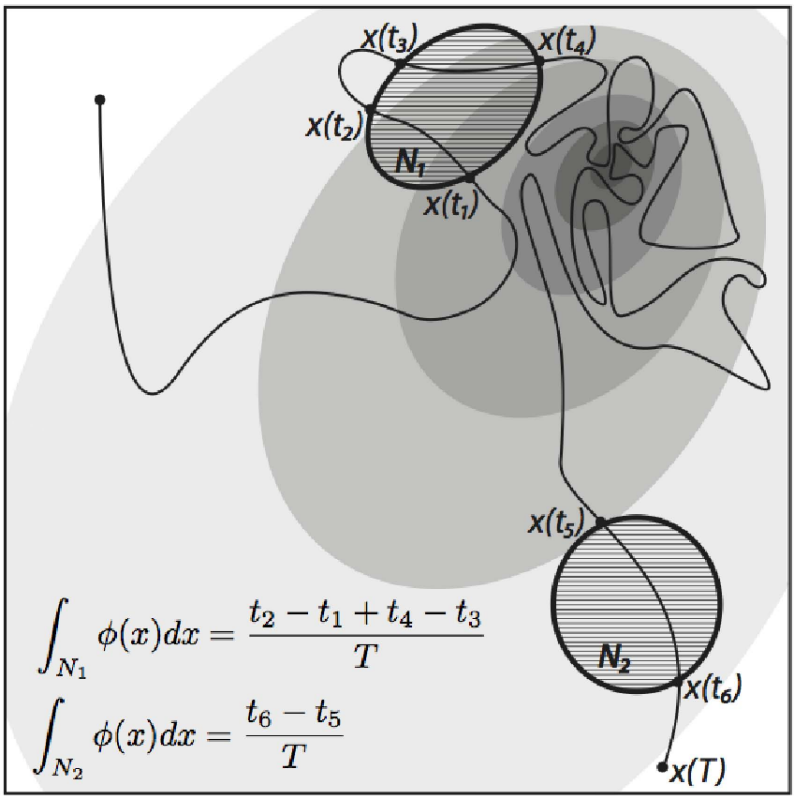



Figure 7: This figure shows an example of an ergodically sampled trajectory. A high number of samples are located in the darker shaded areas that represent areas of higher information density. (Source: [26])

# 3 Learning framework

The goal of this thesis is to validate the feasibility of successfully learning and reproducing a human-demonstrated search for an assembly task. To benchmark the work, the approach is be tested in a Peg-in-Hole task. The following sections explain the steps of the framework necessary to achieve this. To learn the necessary motions it is necessary to first of all provide suitable demonstration (Section 3.1), from which the search space and the forces can be learned (Section 3.2). Using this data, a model of the search can be constructed, from which a search trajectory can be created (Section 3.3). Finally, the learned motions need to be reproduced (Section 3.4).

## 3.1 Providing demonstrations

The first part of this work is to provide the necessary demonstration, in order to learn the search space. As mentioned in Chapter 2.4, searches are needed in assembly tasks when visual information of the environment is unavailable, such as when vision systems can either not produce good images or there is simply no space for them. To imitate this non-existing visual information, the demonstrations need to be performed blindfolded. Thus the only kind of information known by the test subjects as well as the robot are their inertial measurements, i.e. how far they have moved. The demonstration is done on the Peg-in-Hole task, which is shown in Figure 4 (d).

The search and subsequent completion of the task should be performed in one single demonstration as this is an integral part of the research question. It is thus necessary to provide a demonstration that covers as much area as possible, meaning that the search should start as far away from the goal as possible. This section gives an overview of the way the necessary demonstration is conducted. First, the order of the demonstration is presented:

1. The end-effector with the sensor and the peg is placed as close to the border of the search area (black cylinder in Figure 4 (d)), with about $1cm$ of air between the surface and the tip of the peg.

2. The teacher is shown this initial configuration, their hands are placed above the F/T sensor on the robot end-effector, and the recording of states and actions is started.

3. The teacher then closes their eyes and begins the search for the hole.

4. The teacher performs their search until either completion of the task or failure. The demonstration was considered a failure when the peg slid off the surface of the cylinder.

5. If the demonstration was a failure, it is not used. Instead the demonstration begins again from step 1. If the demonstration was successful, the recorded data is used for learning.

The reason the initial configuration of the task is shown to the teacher in step 2 is that the demonstrations would fail too often if the teacher was blindfolded before that and had to guess the initial configuration. The fact that a human cannot tell the exact difference and direction from this initial configuration and instead only estimate a rough guess resembles a system with bad or incomplete vision systems. It is thus justifiable to conduct the demonstration in such a way.

The teacher that demonstrates the task can be any person. Teachers with more knowledge of the task might use that knowledge subconsciously when performing the search, while inexperienced users demonstrate purely their individual search motions. However, a quick trial on users that were not familiar with the task showed that many succeed eventually but fail frequently before that. It was thus decided that the author of this thesis performs the demonstrations, in order to to avoid inconvenient and time-consuming failures during demonstrations.

At this point it is also important to discuss the quality of the demonstrations that were made. For example in [38] they mention that especially users with little to no experience in teaching a robot perform inefficient demonstrations. In the paper the authors propose a visual feedback system to help the teachers. However, since in the case of this thesis the demonstrations were done by the author, who already had knowledge about the system, it can be assumed that the demonstrations were of a high enough quality. Additionally, since the main goal of the demonstrations is to learn search motions, there hardly is a correct or incorrect way to perform demonstrations, as long as the teacher has no visual information about the current state.

Because of the flat surface of the cylinder, a simplification can be made that there is no vertical component in the search trajectories. Thus the vertical dimension of the recorded trajectories and forces are ignored for this part of the work. Similarly, any generated and reproduced trajectories, as well as the forces, have no vertical component. This not only simplifies the approach, but it also leads to great savings in computational time and resources during the later part of the learning. However, some downwards directed force is necessary to guide the peg into the hole. Although it is possible to learn the vertical force from human demonstration, this was not done due to time limitations and therefore a constant value was used during reproduction.

Thus the search only needs to be learned in a 2D environment. However, to do this a suitable coordinate system needs to be chosen. One possibility is the world frame of the robot. While this poses no problem for demonstrations with a single search phase, tasks where multiple searches occur during different parts of the demonstration, however, result in several unconnected search motions. Thus, each search is defined to start at the coordinates (0,0) of a Cartesian 2D search coordinate system called *search frame*. The x- and y-axis of this search frame are defined to align with the ones from the world frame, so that trajectories need only to be translated for a coordinate system transformation between world- and search

frame. This however also means that, depending on the number of search motions during the demonstration, the trajectories in the search frame are diverging from (0,0) in different directions. This in turn means that the model of the search has to account for data that has several distinct parts.

During the demonstrations it is important to record both the forces and the current position of the tool center point (TCP), both of which have two dimensions only, as mentioned above. With this there is now enough information available to build a policy that resembles that of a human. Of course only the time frame during which the search actually occurred needs to be used for building the policy. In case of the peg-in-hole task it means that both the beginning, where the peg moves downwards from its starting position, and the final motion, where the peg slides into the hole need to be cut from the force- and position-data. Since the search is only conducted in the x-y plane, a search phase is considered as such if the velocity in the z direction stays below a certain threshold. In this case the threshold was experimentally determined as $0.035 m/s$. The detected search phase for the demonstrated trajectory can be seen as the orange marked part in Figure 8. This approach also enables extracting search phases for different tasks than this peg-in-hole assembly that might include several distinct search phases, although the threshold might be different. However, this is not the focus of this thesis. Once the position and force data are shortened to the search motions only, the position data needs to be transformed from the world frame to the search frame as mentioned earlier. For this, it is sufficient to just translate the trajectory from its current starting coordinates to the coordinates $(0, 0)$. The data of the recorded forces does not need to be transformed or modified further in any way. This means all the data is now ready to be processed by the learning part, which is explained in the following section.

During the early development of this thesis, other approaches of demonstrations were investigated. For example providing the teacher with ear-protector so as to not let them be guided by the sounds created during the search, as the future reproductions would not have any information about that. However, during a first test phase the ear-protectors were deemed unnecessary, as the sound would be of no help to solve the task. Another idea was to provide two, three or four demonstrations, but this was abandoned in order to stay true to the research question and keep the framework as simpleb as possible.

## 3.2 Learning the search space and forces

With the position- and force-data available in the correct form, it is necessary to build a policy over the complete search space. In order to do this, the search space and the tasks dynamics need to be modelled.

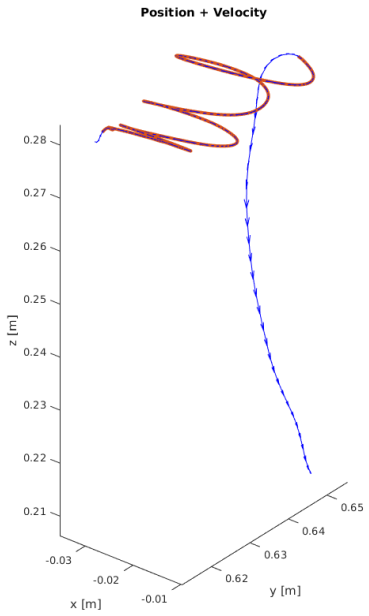



Figure 8: The demonstrated search trajectory used in order to learn a new search. The blue part shows the demonstrated trajectory, while the orange marked part shows only the search motion to be learned.

### 3.2.1 Model definition

As mentioned in Section 2.1.2, a *Gaussian Mixture Model* (GMM) is used to model the search space used by the teacher during the demonstration, i.e. the combination of 2D position and 2D forces. More precisely, the search space explored by the human demonstrator is learned by fitting a GMM to the states and actions during the demonstration. A GMM describes the data as a composition of an arbitrary number of Gaussian distributions where each Gaussian has its own weight, mean and covariances [10]. These GMMs can be used to describe data whose distribution can not be explained well enough by just one Gaussian.
To be able to learn what kind of action leads to which state, information about the current state, the next state, and the interaction that lead from the current to the next state needs to be provided. In the above section the extraction of the position and force data was explained. These two datasets now need to be combined with the information about the next state, which is merely the current position, shifted by one timestep into the future. Now it is possible model the dynamics $p$ of the task, by constructing a joint Gaussian distribution, also known as multivariate Gaussian distribution or multivariate normal distribution, that includes the current states $x_t$ and $y_t$, the next states $x_{t+1}$ and $y_{t+1}$ as well as the interactions $F_x$ and $F_y$:

$$p(x_t, y_t, x_{t+1}, y_{t+1}, F_x, F_y) = \mathcal{N}(x_t, y_t, x_{t+1}, y_{t+1}, F_x, F_y) \ . \tag{2}$$

This is a six dimensional Gaussian distribution, with 6 mean values and a $6 \times 6$ covariance matrix. The higher the dimensionality is, the higher the time and

computational resources will be when finding the best fit for the supplied data. Thus, instead of taking the coordinate values of the current and next position, the vector $(\Delta x, \Delta y)$ from the current state to the next is calculated. This makes the position data location invariant. Not only does this simplify the model, but it also generalizes the learned search motions. Of course this is not possible in environments where similar search motions lead to different results. For example, in an environment with many obstacles, the trajectory of a demonstration depends heavily on which obstacles were hit and which were not. In the case of this peg-in-hole task however, this simplification is absolutely justified. Thus, the dynamics can now be modeled as the following four dimensional joint distribution:

$$p(\Delta x, \Delta y, F_x, F_y) = \mathcal{N}(\Delta x, \Delta y, F_x, F_y) \ . \tag{3}$$

Equations 2 and 3 describe only a single Gaussian. Thus, in a GMM, there are $(K \mid K \in \mathbb{N}_{>0})$ different joint distributions like Equation 3, each with its own relative weight $\pi_k$ $(\sum_{k=1}^{K} \pi_k = 1)$. To find the best fit of all these Gaussians to the supplied data, it is necessary to use an Expectation-Maximization (EM) algorithm that iterates through different fits to find the best one. However, it is important to find the correct number of components in the GMM in order to determine which model to select.

### 3.2.2 Model selection

To be able to find the number of Gaussians in the GMM that fits the demonstrated search space best, it is important to use an objective measure. In general, the more Gaussians there are in the GMM, the better the fit. However, this is because the high number of Gaussians makes it possible to fit precisely to the supplied data, instead of assuming the form of the underlying distribution. This type of behaviour is called overfitting in the machine learning literature [31] and it is not desirable. To combat this, the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) were conceived [11]. While they both find the best potential fit for a dataset, trading off both the goodness of the fit and the dimensionality of the model, the BIC tends to favour lower dimensionalities [11]. Since a lower dimensionality is both easier to understand from a human perspective and computationally less expensive for the construction of models, the the focus is from now on on the BIC. The BIC is defined originally according to [11] [12] as

$$2log(L) - n_p log(n_o), \tag{4}$$

where $log(L)$ is the log-likelihood, measuring the goodness of a fit, $n_p$ is the number of parameters, and $n_o$ is the number of observations. The best model is the one that maximizes Equation 4. However, the BIC is also often written as the negative of equation 4 [13, 14, 15]:

$$-2log(L) + n_p log(n_o). \tag{5}$$

In this case, the best model is the one that minimizes Equation 5, thus when comparing different models for the GMM, the one with the smallest BIC value is

chosen. This is also the method for selecting the model complexity used in the further course of this thesis.

While the implemented algorithm can use a GMM of any dimensionality and size, time limitations of the thesis allowed only for a mathematical model of the state dynamics. Thus a simplification of the GMM needs to be made before a model can be defined, i.e. the number of components needs to be fixed. To do this however, a the BIC value of the different GMM sizes needs to be calculated. As Figure 9 shows, there is a steady trend of the BIC decreasing as the number of components in the GMM rises. However, it should be noted that the actual value of the different components varies only within a range of $1 \times 10^3$. Because of this range, and to keep the model as simple as possible, it was chosen that there is only one component in the GMM. This makes the model of the dynamics a single, four-dimensional joint distribution. However, with the single demonstration as basis for the learning, this simplification is also necessary to avoid overfitting. The resulting model of the search space is as follows:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix}, \; and \; \boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \Sigma_{13} & \Sigma_{14} \\ \Sigma_{21} & \Sigma_{22} & \Sigma_{23} & \Sigma_{24} \\ \Sigma_{31} & \Sigma_{32} & \Sigma_{33} & \Sigma_{34} \\ \Sigma_{41} & \Sigma_{42} & \Sigma_{43} & \Sigma_{44} \end{bmatrix}, \tag{6}$$

where index 1 signifies the force $F_x$, index 2 the force $F_y$, index 3 the state change $\Delta x$ and index 4 $\Delta y$.

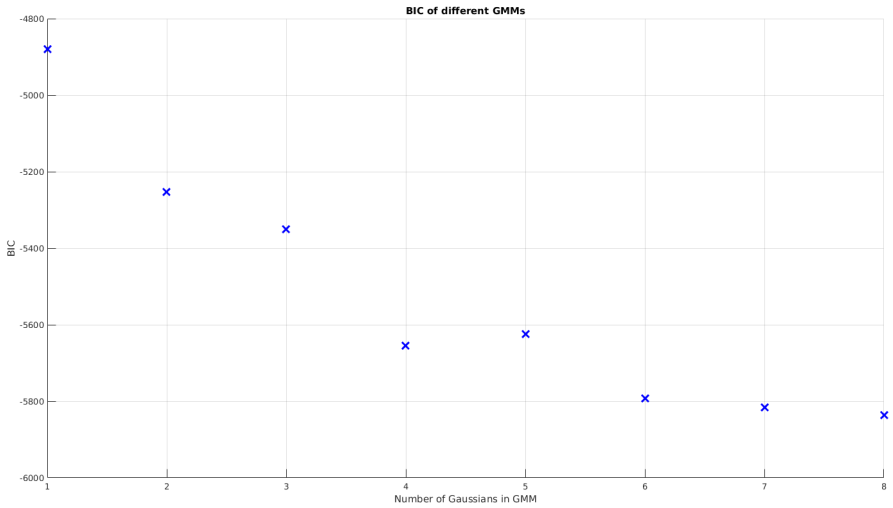



Figure 9: The BIC value decreases with increasing number of components in the GMM. There is no clear number of components that is preferable, as it can be assumed the the shown trend continues further.

### 3.2.3 Prediction of forces

With the dynamics now fully modeled as a four-dimensional Gaussian distribution, it is now possible to predict the forces that are necessary to move from one state to the next. This is achieved by the use of conditional Gaussians. Calculating the conditional distribution is described in [39]. First the four-dimensional vector of states and actions needs to be partitioned in the following way:

$$\boldsymbol{x_{cond}} = \begin{bmatrix} \boldsymbol{x_{cond,1}} \\ \boldsymbol{x_{cond,2}} \end{bmatrix}, \; with \; \boldsymbol{x_{cond,1}} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} and \; \boldsymbol{x_{cond,2}} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} . \tag{7}$$

Accordingly, the mean and covariances from Equation 6 need to be partitioned into the conditional distribution:

$$\boldsymbol{\mu_{cond}} = \begin{bmatrix} \boldsymbol{\mu_{cond,1}} \\ \boldsymbol{\mu_{cond,2}} \end{bmatrix}, \; with \; \boldsymbol{\mu_{cond,1}} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} and \; \boldsymbol{\mu_{cond,2}} = \begin{bmatrix} \mu_3 \\ \mu_4 \end{bmatrix} \tag{8}$$

$$\begin{aligned} \boldsymbol{\Sigma_{cond}} &= \begin{bmatrix} \boldsymbol{\Sigma_{cond,11}} & \boldsymbol{\Sigma_{cond,12}} \\ \boldsymbol{\Sigma_{cond,21}} & \boldsymbol{\Sigma_{cond,22}} \end{bmatrix}, \; with \\ \boldsymbol{\Sigma_{cond,11}} &= \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} and \; \boldsymbol{\Sigma_{cond,12}} = \begin{bmatrix} \Sigma_{13} & \Sigma_{14} \\ \Sigma_{23} & \Sigma_{24} \end{bmatrix} and \; . \\ \boldsymbol{\Sigma_{cond,21}} &= \begin{bmatrix} \Sigma_{31} & \Sigma_{32} \\ \Sigma_{41} & \Sigma_{42} \end{bmatrix} and \; \boldsymbol{\Sigma_{cond,22}} = \begin{bmatrix} \Sigma_{33} & \Sigma_{34} \\ \Sigma_{43} & \Sigma_{44} \end{bmatrix} \end{aligned} \tag{9}$$

With this it is now possible to calculate the distribution of $\boldsymbol{x_{cond,1}}$, with the condition that $\boldsymbol{x_{cond,2}} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$ as a Gaussian distribution:

$$(\boldsymbol{x_{cond,1}} \mid \boldsymbol{x_{cond,2}} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}) \sim \mathcal{N}(\boldsymbol{\mu^*}, \boldsymbol{\Sigma^*}), \; where \tag{10}$$

$$\boldsymbol{\mu^*} = \boldsymbol{\mu_{cond,1}} + \boldsymbol{\Sigma_{cond,12}}\boldsymbol{\Sigma_{cond,22}^{-1}}(\boldsymbol{x_{cond,2}} - \boldsymbol{\mu_{cond,2}}), \; and \tag{11}$$

$$\boldsymbol{\Sigma^*} = \boldsymbol{\Sigma_{cond,11}} - \boldsymbol{\Sigma_{cond,12}}\boldsymbol{\Sigma_{cond,22}^{-1}}\boldsymbol{\Sigma_{cond,21}} \; . \tag{12}$$

For our needs, the covariance matrix $\boldsymbol{\Sigma^*}$ is not necessary, only the mean $\boldsymbol{\mu^*}$, which now represents the mean of the forces $\begin{bmatrix} F_x \\ F_y \end{bmatrix}$ that are predicted, given the states $\boldsymbol{x_{cond,2}} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$.

## 3.3 Learning the search trajectory

To create a search trajectory that is similar to the one demonstrated by a human, it is necessary to learn the area of the search frame, where the teacher demonstrated the search. Since the location invariant version of the position data cannot provide

information about the density of positions within the search frame, it is necessary to fit the distribution over the non-location-invariant version of the position data, which was described at the very end of Chapter 3.1. For this, a bi-variate Gaussian distribution is fitted over the 2D position data in the search frame and is then optimized with an EM algorithm, similar to the modeling of the 4D search space in the previous section. Again, the reason for choosing a single Gaussian instead of a multi-component GMM was that this approach is on one hand simpler and on the other hand reduces the risk of overfitting.

### 3.3.1 Calculating the itinerary

The next step in order to create a trajectory from this distribution is to sample a number of points from it. However, the question of which number of sample points to choose is not a trivial one. While more points inevitably lead to a higher coverage rate of the search space, the resulting trajectory is also longer, which in turn may lead to longer runtimes in the end. What kind of number is best suitable for reproduction is discussed in Chapter 5.2.

After the points are sampled, a route needs to be developed that goes through each one of these points to cover the distribution in one go. In order to reduce time and resources during reproduction, the generated trajectory needs to be as short as possible. Thus, an itinerary is created that lists the sampled points in such an order that the resulting route is as short as possible. The problem of finding an optimal path through $n$ points is also known as the Travelling Salesman Problem (TSP). To create such a path, a MATLAB code solution by Joseph Kirk is used [42]. This solution tries to find a route that is as short as possible by using a Genetic Algorithm (GA) to calculate the shortest distance between a fixed starting position and the sampled points. Obviously, the more points there are, the exponentially longer the algorithm needs to create the optimal route. This means that it is best to make due with as few sample points as possible. The start of the optimal path is always set to the coordinates (0,0), so that each search trajectory starts at the point where the search needs to start in the reproduction.

### 3.3.2 Smoothing the route

However, this route only connects the points in the search space. In order to provide a smooth trajectory for reproduction, this discrete path needs to be filtered and smoothed. To achieve this, several different approaches were investigated. The first approach was to use the Spline Fitting toolbox in MATLAB to fit a natural spline to the optimal route. The resulting trajectory however passes through each point. This means in the areas where the points lie close together, the trajectory becomes jittery. Since these closely together lying points are within a few millimeters of each other, the possibility of filtering the trajectory before applying a smoothing spline to it was investigated. This provided a smooth path through even the densely sampled

areas and avoid undue stress on the robot's actuators. In the following, two methods are presented that can provide the necessary filtering.

The first method is achieved by applying a Savitzky-Golay filter [40]. As can be seen in Figure 10, this filter allows for a good amount of smoothing, while still preserving the features of the original function. Basic filters like a moving window do not allow for such feature-preservation. However, depending on the choice of degree and window-length of the Savitzky-Golay filter, the preservation of features can be traded off for more smoothness. Nevertheless, after the data is filtered, the resulting trajectory still consist of the same number of points. But since these points are now filtered, a natural spline can easily be fitted to the points and results in a continuous and smoothed trajectory.

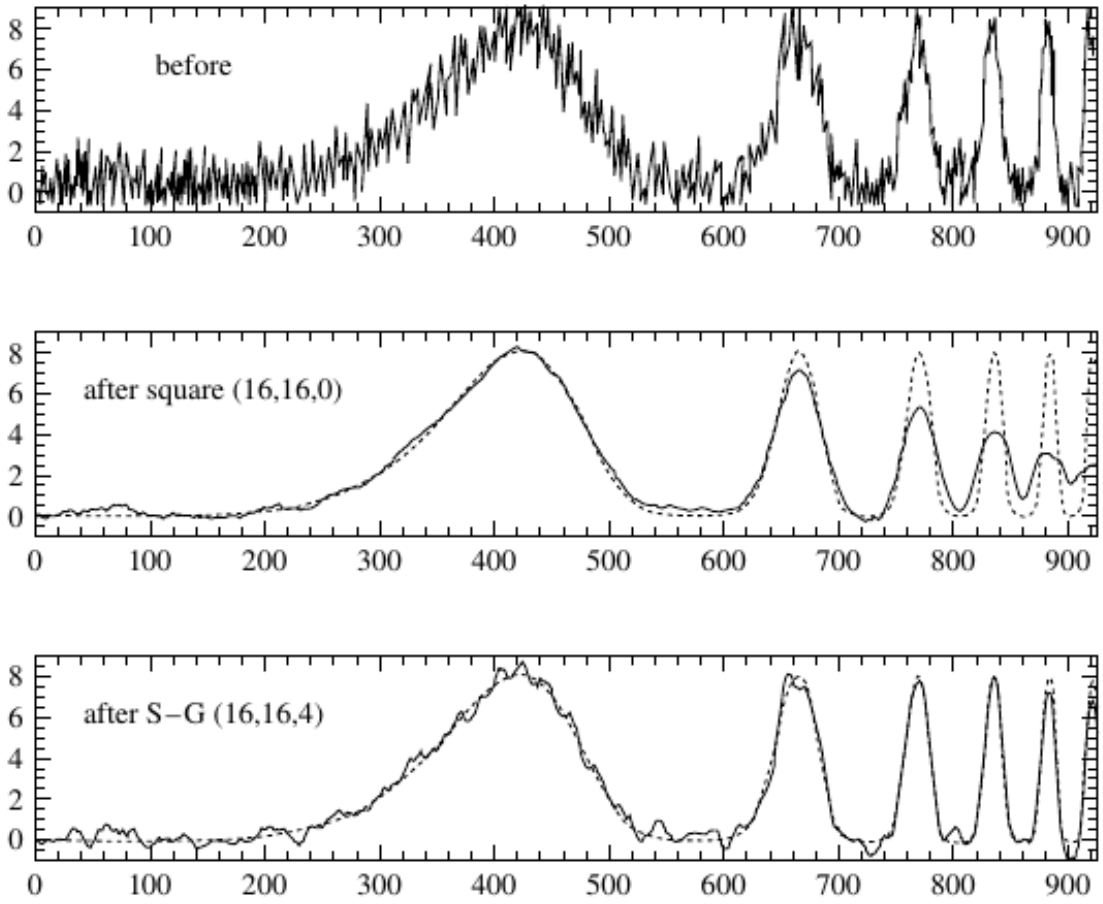



Figure 10: This figure shows the advantages of using a Savitzky-Golay Filter. The upper third shows an example of a noise-afflicted function with bumps that get continuously tighter. The middle frame shows the results of filtering the above data with a basic moving window filter, which smooths the function but looses features such as the height of the narrower bumps. The bottom frame shows the data filtered by a Savitzky-Golay filter, where the the data is smoothed to a lesser degree but the features of the original function are preserved. (Source: [40])

Another method that was investigated was the SMOOTHN-function provided for MATLAB by Damien Garcia [43]. This method smooths the data using a *"fast robust version of a discretized smoothing spline [that is] based on the discrete cosine transform (DCT) [and] allows robust smoothing of equally spaced data in one and higher dimensions"* [44]. The advantages of this method are the robustness against outliers in the data, as seen in Figure 11. The data used in this thesis is of course only consistent of sample points, and as such there are no outliers per se. Nevertheless this method might prove useful, as the robustness can help also reduce motions in densely sampled areas. Finally, similar to the above approaches, a natural spline needs to be fitted to the resulting smoothed path as well to provide for a continuous trajectory function.

Before this trajectory can be deployed for reproduction however, the expected environmental forces need to be generated. To be able to do this, the trajectory needs to be made location invariant again (compare Chapter 3.2). However, the trajectory functions returned after the filtering and smoothing process are continuous. This means that, to create the vectors between the points of the trajectory, the continuous trajectory function first needs to be subsampled, i.e. a discrete set of points needs to be sampled from the trajectory function with a certain distance between each point. These discrete points still need to describe a smooth path however. Thus the correct distance between successive points needs to be chosen carefully. With the discrete set of points available, the location invariant data can be created, which is then fed into the conditional Gaussian (see Equation 10). After producing both the trajectory and the corresponding forces, the search motions are ready to be deployed to the robot for reproduction.

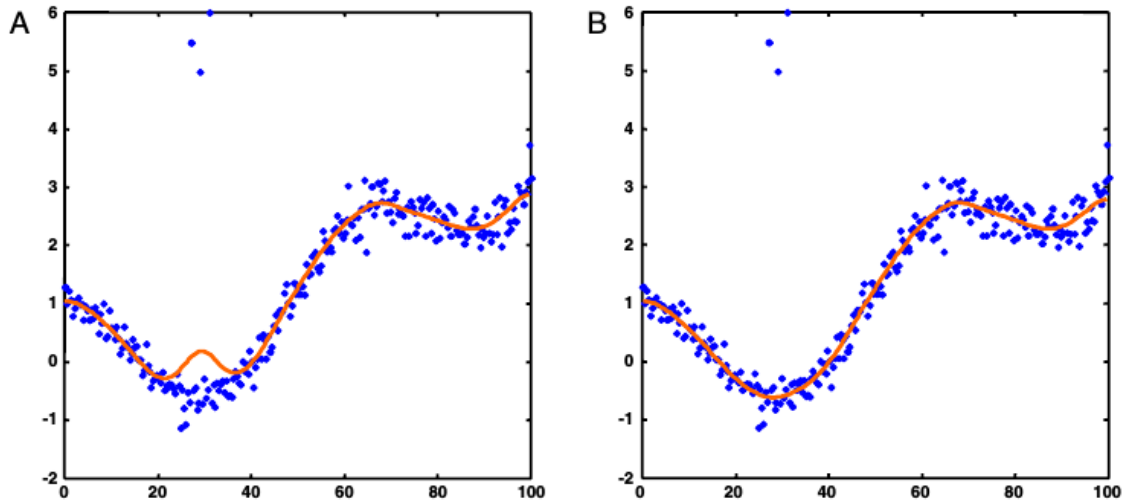



Figure 11: This figure shows the advantages of the discretized smoothing spline presented by [44]. The left frame (A) shows that outliers may distort trajectories from regular filtering methods. The use of the proposed method however is robust against such deviations, as seen in the right frame (B). (Source: [44])

## 3.4 Reproduction

To reproduce a search similar to that of the demonstrated one, it is necessary to have a controller on the robot that executes and determines the forces that are necessary to make the robot move in the intended fashion. For this, an impedance controller was developed, that utilizes the generated trajectory and the predicted forces. Additionally a less complicated force controller is designed as a comparison as well.

### 3.4.1 Impedance Controller

The *impedance controller* is the main work of this thesis and also the most refined controller that was designed in the course of this thesis. In general, impedance control provides the user with a dynamic interaction between robot and environment. The *stiffness* of the controller gives a measure of how high or low the positional accuracy should be, compared to the level of contact forces between the robot and the environment. A low stiffness value of for example $1\frac{N}{m}$ means it is easier to move the robot from its desired position, as only one Newton of force is applied to the robot for every meter distance from the desired position. This means that an impedance controller is suitable for the execution of compliant motions (see Chapter 2.2), assuming that the stiffness parameter is set accordingly. The impedance controller can also be combined with an additional force-feed-forward component, as can be seen in Figure 12, which is the configuration used in the main work of this thesis. The impedance controller used in this thesis is in principle a spring-damper system:

$$F_{control} = Ke_x + B\dot{e}_x + F_{feed-forward} \ , \tag{13}$$

where $e_x$ is the error in current position and desired position, $K$ is the stiffness coefficient, which tells how much force is required to move by $1m$, $\dot{e}_x$ is the derivative of the error, and $B$ is the damping coefficient, which tells how much force should be used to dampen the motion. The additional $F_{feed-forward}$ term stands for the forces that are fed forward.

Because of the inclusion of both feed-forward forces and the positional error, using an impedance controller allows for following a trajectory while being compliant to the environment, assuming the control parameters are configured correctly. This however can also be of a disadvantage, as it is usually a laborious and time-intensive process to figure out the correct settings for stiffness and damping. The addition of the feed-forward forces is not complicated however, while adding another layer of control. These advantages made the impedance controller with force-feed-forward the prime candidate for the experiments used to validate the work of this thesis. Another advantage of this impedance controller is that it is able to move in free space as well if the positional error $e_x$ in Formula 13 is big enough, which is another reason why this controller was chosen as the main focus. However, in order to compare this controller, another alternative is presented that signifies a simpler approach.

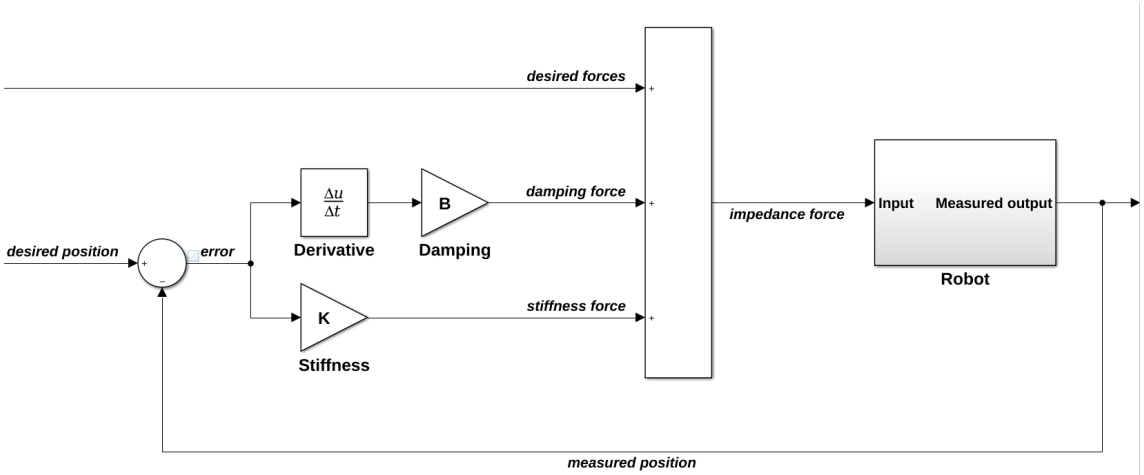


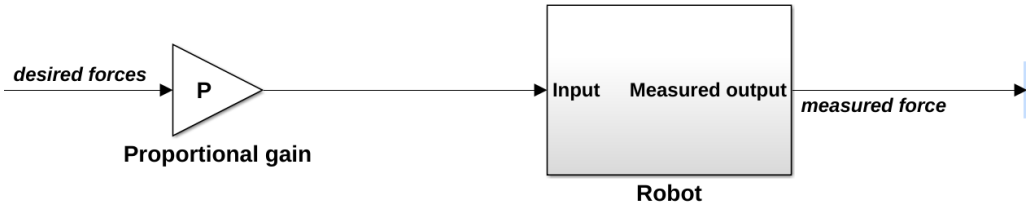

Figure 12: The impedance controller with force-feed-forward is the most refined controller of those designed during the course of this thesis, and is also the one successfully used for the experiments.

### 3.4.2 Force Controller

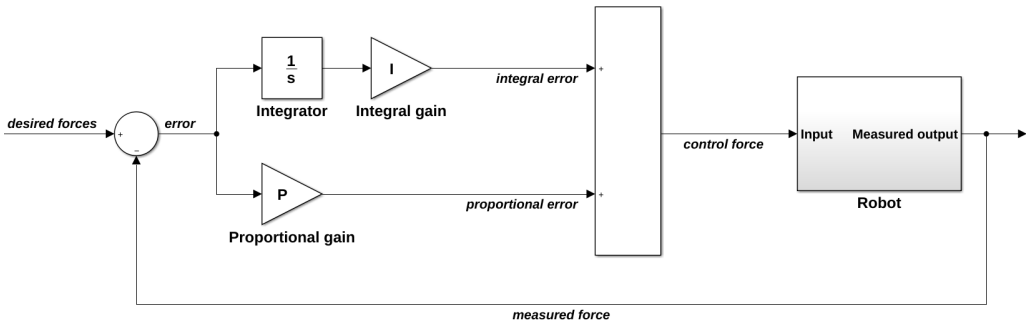
The *Force-Feed-Forward Controller* that was developed to provide a basic alternative to the impedance controller merely plays forward the forces it receives directly to the robot. It should be noted at this point, that the forces learned from the demonstration are in toolframe-coordinate-system, and thus can immediately be forwarded to the controller. They however need to be negated, as the forces recorded during the demonstration are those that result from the environment, which mean they are friction forces in this setting. Additionally, they may be multiplied by a proportional gain. A schematic of the controller can be seen in Figure 13 (a). However, since a force-feed-forward controller might not produce satisfying results, a force-feedback controller was designed. This force-feedback controller, which can be seen in Figure 13 (b), is a PI controller. This means it takes the difference between the desired force and the measured force, multiplies it by a certain gain. Additionally, the error is integrated and subsequently multiplied by an integral gain as well. After the two values are added, they are commanded to the robot. Both for the force-feed-forward controller as well as for the force-feedback controller, the desired force is updated with each timestep ($10ms$), which means that the feedback of the force-feedback controller lags behind by one timestep, since the measurements are outdated by one timestep.

The advantages of using only a force-feed-forward controller is that it is both easy to understand, as well as quick to develop, program and debug. However, this form of force controller does not have any feedback, and thus cannot adapt to the environment and has to rely on the correctness of the programmed desired values, in this case the generated forces. However, because of the lack of feedback, it is not necessary to have the ability to sense and measure forces on the robot, i.e. no

external or internal force sensor is necessary. On the other hand, the force-feedback controller possesses the ability to adjust to the environment, as the error between the measured forces and the desired ones commands the resulting forces. This in turn means that a sensor of some sort is required to be on or in the robot. This controller is still simple nevertheless, and can easily be programmed and debugged, as the only setting that needs to be adapted to the environment is the proportional gain that is multiplied with the force-error. In the end however, both force controllers have to rely nevertheless on a precise and accurate execution of force-commands to the robot, as well as – for the force-feedback controller – high-precision measurements.




(a)




(b)

Figure 13: The force controllers that serve as a simpler alternative and comparison to the impedance controller. (a) shows the force-feed-forward controller, while the controller in (b) is working with force-feedback.

# 4 Setup

In the previous chapters, it was described how the demonstrations are made and how a search can be reproduced from those human demonstrations. In order to evaluate how well the search reproductions work under different circumstances, a series of experiments is made, which are explained in the second part of this chapter. To do this however, the necessary infrastructure first needs to be set up. To implement, execute and evaluate the algorithms described in the previous chapter, an advanced hardware- (HW) and software (SW) environment is necessary. The existing architecture of both HW and SW are described in the following sections, and an overview over the different components and their connections can be seen in Figure 14.

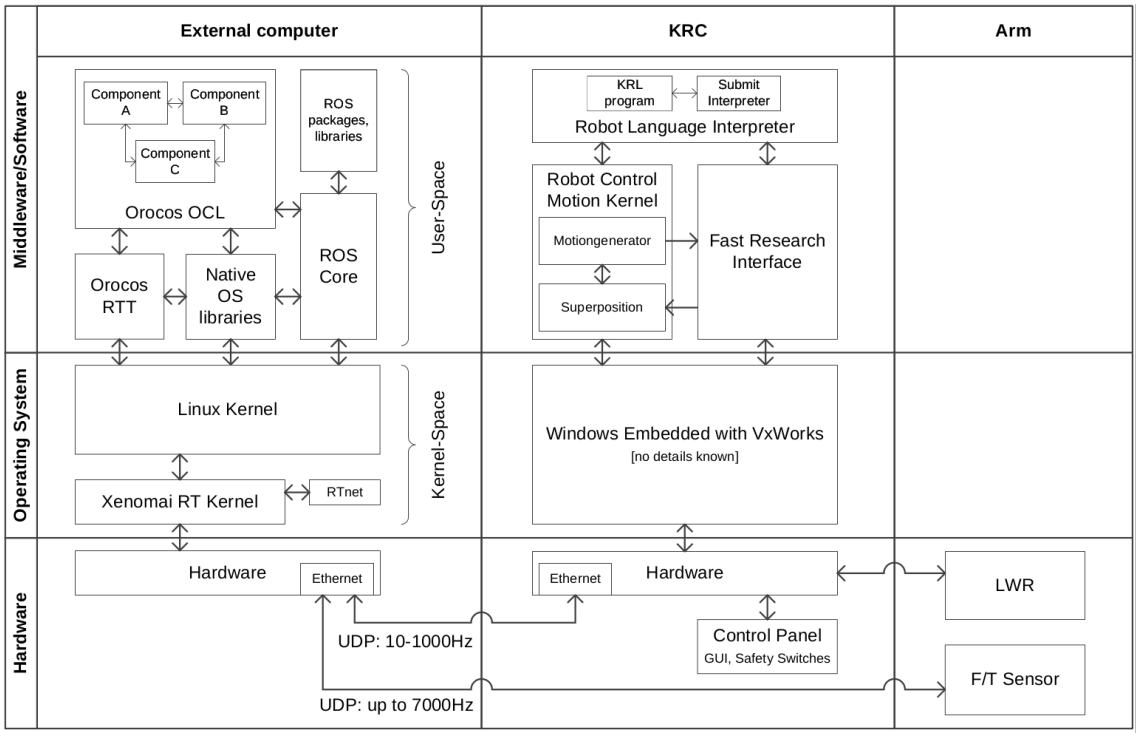



Figure 14: An overview of the hard- and software used in this thesis and their interfaces to one another. (Source: [45])

## 4.1 Hardware

The hardware used in this thesis consists of several components, which can be seen in Figure 15: the KUKA LWR4+ lightweight robotic arm, the KUKA Robot Controller (KRC), an ATI Mini45 F/T sensor and an external computer. The KUKA LWR4+ was developed by KUKA Roboter and the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR) [48]. With a weight of $15kg$, this robotic arm is able to handle payloads up to $7kg$. The robot possesses seven different rotational joints, or degrees of freedom (DOF), which means it has one redundant axis. This allows it to complete dexterous movements, while avoiding singularities. These two

features make the robot predestined to be used for the Kinesthetic Teaching method described in Chapter 2.1.1, as both the low weight as well as the dexterity improve the human interaction with it immensely. The arm also has the ability to measure forces and torques, which not only allows for an active gravity compensation, i.e. it is able to compensate for its own weight, it also makes the use of complainant motions possible. These are two fundamental necessity for allowing demonstrations to be made and the reproductions to be executed.

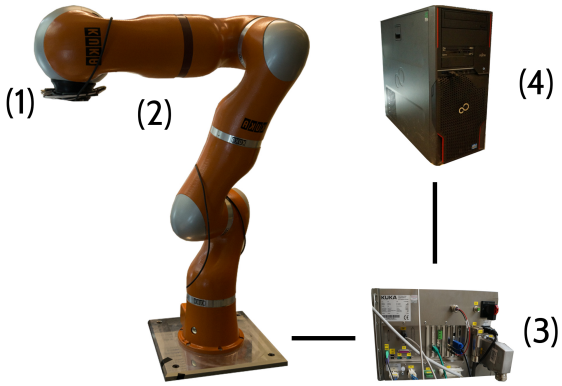



Figure 15: This figure shows the KUKA LWR4+ lightweight robotic arm (2) with its tool flange (1), where tools or the F/T sensor can be attached, as well as the KRC (3) and the external computer (4). (Source: [46], adapted)

The KRC, which is directly connected to the KUKA arm, is responsible for the safe and controlled operation of the robot. Besides the necessary power supply, safety circuits and other necessities for the robot, it features a Windows Embedded interface, which allows the user to operate the robot, as well as write and execute programs written in the KUKA Robot Language (KRL).

In addition to the force- and torque measurements of the internal sensors, an external ATI Mini45 F/T sensor can be attached to the tool flange of the robot arm that allows for high precision measuring of external forces and torques acting on it. While the robot itself can measure external forces during the reproduction, this sensor is necessary to record the interaction forces of the environment without including the forces the human teacher used during the demonstration. The sensor is calibrated so that it can measure forces up to $290N$ in x- and y-axes, $580N$ in z-axis all with a resolution of $\frac{1}{8}N$. It can also measure torques up to $10Nm$ with a resolution of $\frac{1}{376}Nm$ in x- and y-axes, and with $\frac{1}{752}Nm$ precision in z-axis [50].

Since the KRL interface is not sufficient for most tasks requiring in- and output of sensor-values, an external computer is needed to interface with the robot. To achieve this, an eight-core computer with Ubuntu 12.04LTS and a Xenomai realtime kernel is connected via Ethernet to both the KRC and the F/T sensor. This computer hosts most part of the software needed to conduct the experiments, which is explained in

more detail in the following Section 4.2. One requisite of the computer is that it complies with the strong realtime requirements that are associated with the control of robots. In addition to the aforementioned realtime kernel, this is also supported by two extra network cards that are used for communication with the robot and the sensor.

As mentioned before, the experiments are conducted with the peg-in-hole setup mentioned in Chapter 2.3. The peg is a 80.20$mm$ long plastic cylinder with 16.35$mm$ diameter. One tip of the peg is rounded to a hemisphere, the other is attached to the F/T sensor, which itself is also attached to the robot (see Figure 16). The cylindrical hole that the peg needs to be inserted into is 85.11$mm$ deep, 16.70$mm$ in diameter and located on a plastic cylinder with 128.38$mm$ diameter. Because the end of the peg is not perfectly hemispherical, but a bit ablated, the peg can be expected to slide down the hole within a distance from the hole's center of about 3 to 5$mm$. The effective goal area thus is a circle of roughly 6 to 10$mm$ diameter around the center of the hole. Note that even though during all the demonstrations and the experiments, the orientation of the peg was fixed, during reproduction, the stiffness of the rotational axis was set to $100\frac{Nm}{rad}$, which does allows for small changes in orientation of roughly 1-2 degrees in each direction.

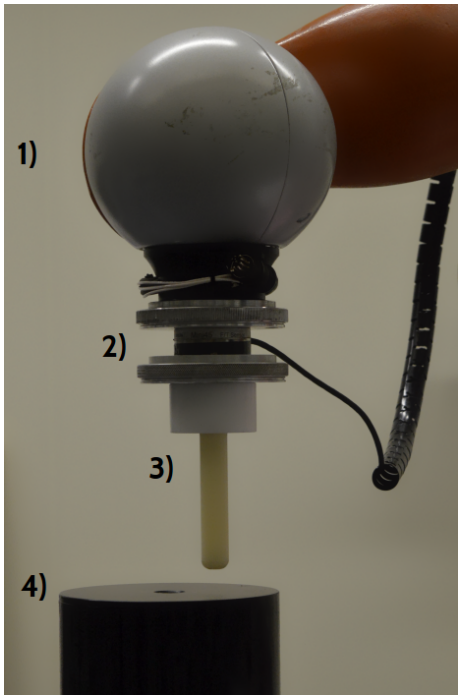



Figure 16: An overview of the peg-in-hole setup used for this thesis. The F/T sensor (2) is directly attached to the tool flange of the KUKA (1), so that when the user guides the robot by grabbing it on the orange surface, the environmental forces can be measured. Attached to the F/T sensor is the peg tool (3). A bit below that is the black cylinder (4) that serves as search area with the hole.

## 4.2 Software

In order to provide the user with an easy to use interface to the robot, a framework was developed that allows easy access to the controllers, sensors and actuators of the robot. This interface is called Fast Research Interface (FRI) and was developed cooperatively by both the DLR and KUKA [49]. The FRI connects the external computer with the KRC over the User Datagram Protocol (UDP) and supports rates of up to $1kHz$ for sending messages between the two parties. With this interface it is possible to use the external computer to command the different internal controllers of the KUKA robot and to set their parameters. It also allows for directly controlling and measuring the desired position, velocity or forces/torques in both the Cartesian as well as the joint space.

### 4.2.1 OROCOS interface

For the user to make use of the FRI however, another interface is necessary. The Open Robot Control Software (OROCOS), combined with the Robot Operating System (ROS) provides the user with this interface, with OROCOS satisfying also the system's realtime requirements. In the OROCOS environment, there are two distinct elements that are important for the user: components and deployers. OROCOS components function as a realtime thread that, after it is initialized, gets called at certain intervals or by external events. They are usually written in C++ and contain the parts of the program that send commands to the robot's controllers. All the work that relates to controlling the robot in this thesis is done in such a component. The second important OROCOS element is the deployer. These files, written in the OROCOS Programming Script (OPS), are necessary to include the different OROCOS components and they also specify when these components should be started and called. The deployer is also responsible for creating the necessary communication between the different components, as well as communication to the outside world, specifically the streaming of measurements and data from the robot and the OROCOS components to ROS topics. Besides the use of ROS topics to enable the user access to easily readable data, ROS is not used elsewhere in this thesis.

An overview of the different components used to control the robot can be seen in Figure 17: The FRIServer enables communication between the KRC and the different components. It transfers for example the desired forces and measured positions. The KUKACommander is supplying the functionality necessary to control the robot, which includes the configuration of the impedance controller for instance. The KUKACommanderROS component serves as an interface between ROS nodes and the KUKACommander, but since ROS is not be further used in this thesis, that part is neglected. Lastly, the FTSensor component provides the other components with the measurements from the F/T sensor.

It is possible to control the KUKA LWR4+ via three different controllers: position control, joint impedance control and Cartesian impedance control. However, the

compliance and force control required for this thesis stems from a impedance controller programmed in an OROCOS component. This impedance controller commands the desired forces resulting from the feed-forward data as well as the impedance control. In order to control the robot with these forces, the internal Cartesian impedance controller was chosen, which is presented in [49]:

$$\tau_{cmd} = J^T(k_c(x_{cmd} - x_{msr}) + D(d_c) + F_{cmd}) + f_{dyn}(q, \dot{q}, \ddot{q}) \ , \tag{14}$$

where $\tau_{cmd}$ are the torques commanded to the robot joints, $J^T$ is the transposed Jacobian that transforms the Cartesian forces into joint torques, $k_c$ is the stiffness value, $x_{cmd}$ and $x_{msr}$ are the desired and measured positions, $D(d_c)$ is the damping term, $F_{cmd}$ are the desired forces and $f_{dyn}$ the dynamic model of the robot.
However, since the impedance control is actually done by the OROCOS component, only the $F_{cmd}$ term is used in this internal controller. That means, no desired position $x_{cmd}$ is sent, as well as the stiffness parameter $k_c$ is set to zero for the three translational axes and to $100\frac{Nm}{rad}$ for the rotational axes. The damping parameter is set to one for all axes. The desired forces $F_{cmd}$ are those coming from the impedance controller implemented in the OROCOS component (see Chapter 3.4.1, Formula 13).

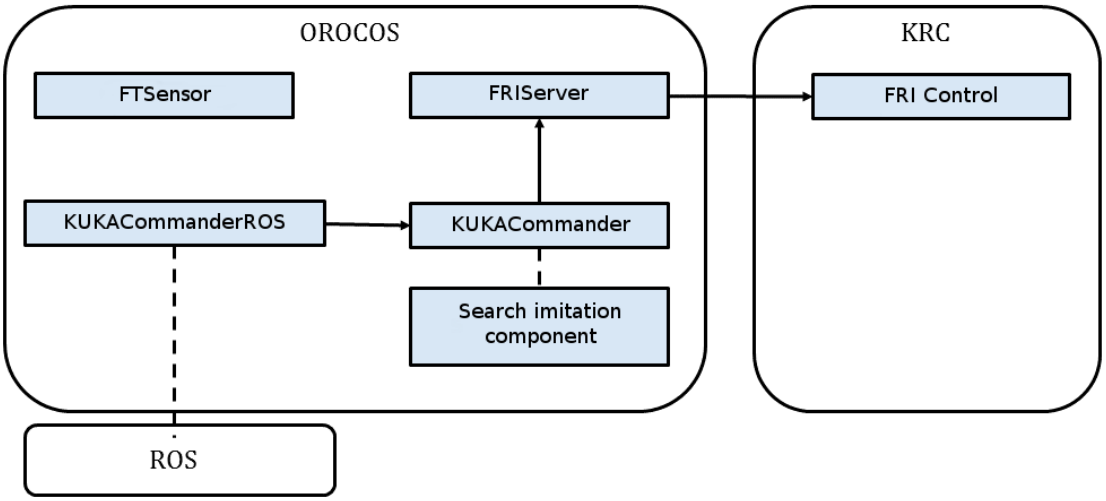



Figure 17: An overview of the different OROCOS components and their connections to the outside world. (Source: [47], adapted)

### 4.2.2 Search imitation component

The search imitation component (see Figure 17) includes all the necessary programming to imitate the search demonstrated and learned earlier. This component was originally created by [47] and modified in order to support the imitation of the search motions. To be able to do this, the generated trajectory from Chapter 3.3 and the predicted forces from Chapter 3.2.3 needed to be supplied in the form of a CSV file. At the start of the component, those files were read and the trajectory points and forces were copied to an internal array. The stiffness and damping parameters of the

KUKA's internal impedance controller (see Equation 14) were then set to zero and one respectively and the parameters for the implemented controller were set so that the stiffness of the x- and y-axes was $1000\frac{N}{m}$, whereas the vertical z-axis has zero stiffness, which makes it completely compliant. The damping was set to $60\frac{Ns}{m}$ for x-, y- and z-axes. Then a value for the force in z-direction was set, which always acts on the tool until the task is done. Since earlier demonstrations showed that the demonstrated level of vertical force was continuously level, a constant value of $10N$ was set, which is similar to that recorded during an earlier human demonstration. It is assumed that once the search imitation component is started, the search should be reproduced. Thus the Cartesian position in worldframe coordinates is saved in the beginning. It is used to convert the generated trajectory points from the search frame to the worldframe by simply adding it. Then, once the component was started, the peg tool was slowly lowered until it touches the surface. Once this happened, the impedance controller started and went through the set of generated trajectory points and forces step by step each iteration ($10ms$). The calculated forces were then sent to the KUKA internal impedance controller, which results in the actual motion of the robot. This is the method which was used to validate the learned search motions in the following experiments.

# 5 Results

This Chapter investigates the effectiveness of the different parameters and configurations that the learning framework can possess. During the design of the learning framework in Chapter 3, many different parameters and configurations were presented. Since any of those parameters can influence the outcome of this thesis, a series of questions was gathered:

1. How does the starting position influence the outcome of the experiments?
2. What kind of effect does the number of sampled points have?
3. What effect do the different ways to generate trajectories have?
4. In which way does the choice of controller impact the outcome?
5. How do the generated forces influence the results?
6. In what way does the randomness of the sampling influence the success of the algorithm?

The answers to these questions are critical for the choice of the final experiment. After these answers are found and discussed thoroughly, a final experiment is conducted. This experiment shows if a search strategy can in fact be learned by just one single human demonstration.
The following sections are divided into several parts that all relate to the questions posed above. Section 5.1 answers question 1. Section 5.2 gives an answer to question 2. Question 3 is explained in Section 5.3. Question 4 is discussed in Section 5.4, where question 5 is evaluated in Section 5.4.2. Finally and most importantly, the gathered answers are used to evaluate the goal of this thesis in Section 5.5, which also answers questions 6.

## 5.1 Influence of starting position

Since this thesis investigates the feasibility of using only one demonstration to teach search motions to a robot, the way this demonstration is made is of great importance. Since the location of the starting position is an elementary part of the demonstration, its effects are especially interesting. Because the distribution that was chosen to model the task is fitted to this one solitary demonstration, the implications of that fit are investigated thoroughly. The chosen distribution is a single 2D Gaussian distribution. This is illustrated in Figure 18 as a red ellipsoid that covers the area within $2\sigma$ from the center of the distribution. This area covers roughly 86% of the distribution [41]. The demonstration data starts off at coordinates $(0, 0)$ and is marked as blue crosses in the Figure. The end of the demonstrated trajectory – i.e. the goal – lies outside of that red ellipsoid, which means that the chances of randomly sampling near this area is rare. This is important since the trajectory learned from

that distribution is created by sampling points from it (see Chapter 3.3.1). Because the Gaussian distribution has its center where the most points were demonstrated, it will be roughly half the distance away between goal and starting position. It is thus imperative that the starting position of a demonstration is always a worst case example. This way, during normal searches, the goal location will on average end up closer towards the center of the Gaussian and thus have a higher chance of being covered by the generated trajectory.

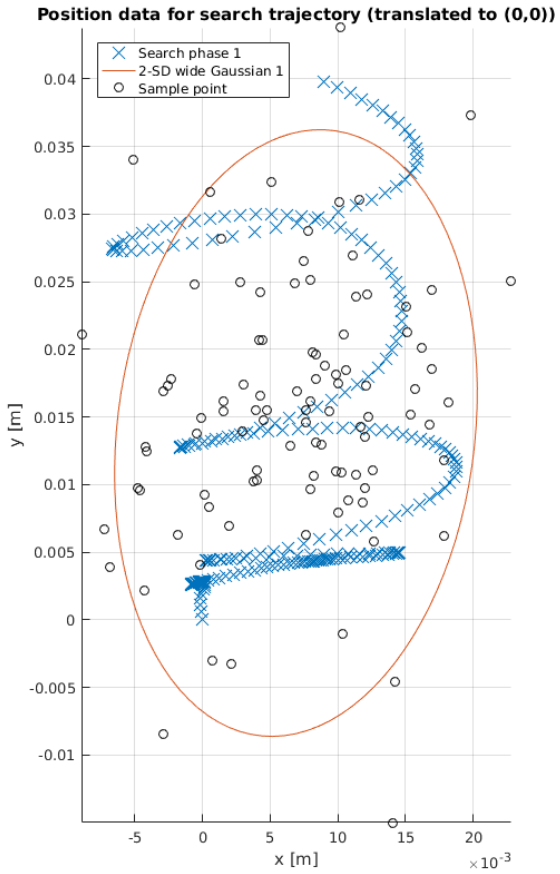



Figure 18: The red ellipse shows the outline of the 2D Gaussian distribution that was fitted to the blue position data at a width of $2\sigma$, which covers roughly 86% [41] of the distribution. The search always starts at the coordinates (0,0). Thus the hole is the upper end of the trajectory. The black circles signify positions that were randomly sampled from the distribution.

To get an idea of the implications of the starting position in the actual experiments of Section 5.5, a real trajectory was evaluated. For this, the impedance controller designed in Chapter 3.4.1 and evaluated later in Section 5.4.1 was used to create a single trajectory. For that particular reproduction, the starting position was moved more towards the goal area. Figure 19 shows that the hole nevertheless is far from the center of the Gaussian, and there are not many sample points around it, especially if compared with the area near the center. What can be gathered from this observation

is that, up to a certain degree, the goal area does not have to be directly in the dense area of the probability distribution to have a good chance of being covered. In addition, from the examination of the trajectory in Figure 19 it can be observed that the effective goal area seems to be around 4 to 5$mm$ in diameter, which is a bit lower than expected in Chapter 4.1. It can be concluded that thus the task is even more difficult than expected. The other key observation made from the figure is that the starting location of the search needs to be moved closer to the goal. Figure 19 proves this rather well, as it can be seen that the goal is, viewed in the search frame, slightly to the left of the starting position by about 5$mm$. Even with the starting position moved closer towards the goal, because of this 5$mm$ difference, the goal is far away from the mode of the Gaussian. It can be seen that the goal (the end of the red trajectory) is just outside of the blue ellipsoid which signifies the $2\sigma$ distance

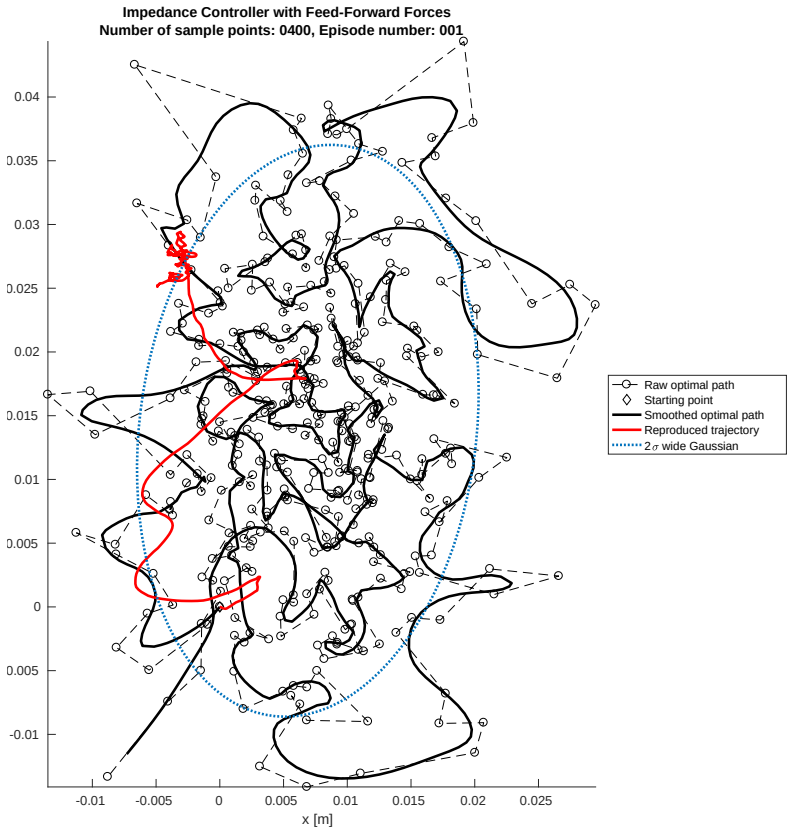



Figure 19: A successful episode with 400 sample points. While the reproduced trajectory (red) differs from the smoothed trajectory (solid black), the goal is found early through the trajectory. Notice that the goal is located up and slightly to the left of the starting position.

to the mode of the Gaussian. It is also clear to see that the reproduced trajectory is not following the desired path accurately. This issue is discussed more in detail in Chapter 5.4.4, as this behaviour is caused by the impedance controller.

## 5.2 Impact of sample sizes

In Chapter 3.3.1 it was hypothesized that the number of sampled points can have a big effect on the outcome of the final experiments, as more sample points cover a greater area. However, since the underlying distribution is a Gaussian (see Equation 6), most of the sampled points will be closer to the center. This is illustrated in Figure 20, where six different sets of sample points were generated. The number of sample points ranges from 100 to 600 in intervals of 100. As can be seen from these figures, the higher the size of the sampling set is, the denser the area inside the red ellipsoid is sampled. The red ellipsoid represents the area that is within two standard-definitions of the distribution, i.e. about 86% [41] of it. It is observed that even with increased sampling rates, the chance of sampling a point outside the red ellipse is still small. This is especially concerning since it was mentioned in the beginning of Chapter 3.3 that during the original demonstration of the task the goal was located far outside of that border (see Figure 18). This means unless the starting position is moved closer towards the goal, the chance of sampling a point near it and thus covering it is almost zero. Therefore it is of great importance to always demonstrate the worst possible starting position. This way the starting point of the average search is closer to the goal, which then has a higher chance of being covered by the sampled path. If the goal is closer towards the center of the distribution, the sampling rate indeed affects the chance of covering goal with the samples. This assumption is validated experimentally in Chapter 5.5.

Another critical point is the method to calculate the shortest path through these sampled points. Naturally, the higher the number of sampled points is, the more

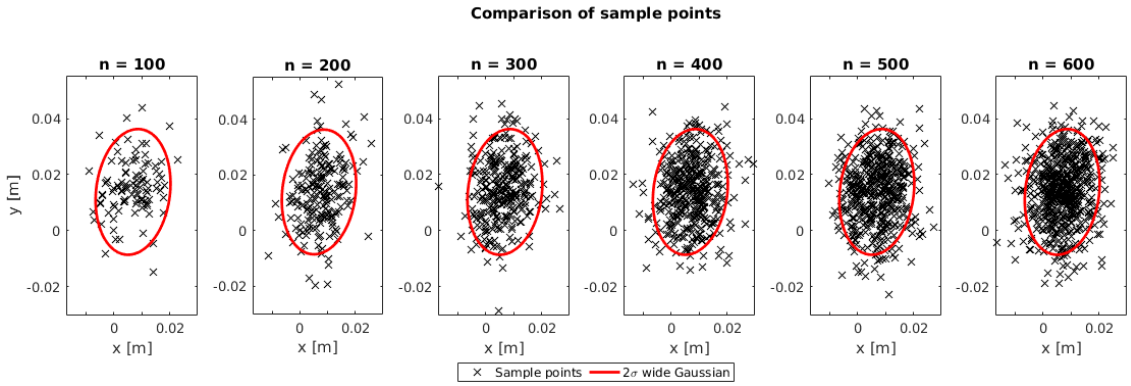



Figure 20: The higher the number of sampling points, the longer the generated trajectory is. This increases the coverage and thus possible success rate, however it also increases the runtime of the trajectory. Note also that even for 600 sample points, there are few points outside of the $2\sigma$ wide area of the Gaussian (red ellipse).

resource- and time-intensive the calculations will be. Since the approach proposed in this thesis is however supposed to be simple and straight-forward, the user should not feel overwhelmed by the time required to calculate the shortest route. Thus, a hard limit on the runtime of the algorithm is enforced. Nevertheless, the higher the sample size, the more inefficient the path is. This means the number of samples can not be chosen arbitrarily high to provide best possible coverage.

## 5.3 Effects of trajectory generation

Chapter 3.3.2 described how a raw trajectory can be transformed into a filtered and smoothed version, which is necessary to allow for a continuous motion of the robot actuator. During that Chapter, two aspects were mentioned, where it is possible for the user to make different choices:

1. The filtering and smoothing method.
2. The rate of subsampling.

The effect the different choices can have on the outcome are possibly far reaching. Thus, in this section, all the possible choices and their effects are explained, evaluated and discussed.

Chapter 3.3.2 presented the user with three methods of converting the desired itinerary into a smooth continuous motion and they all have their advantages and

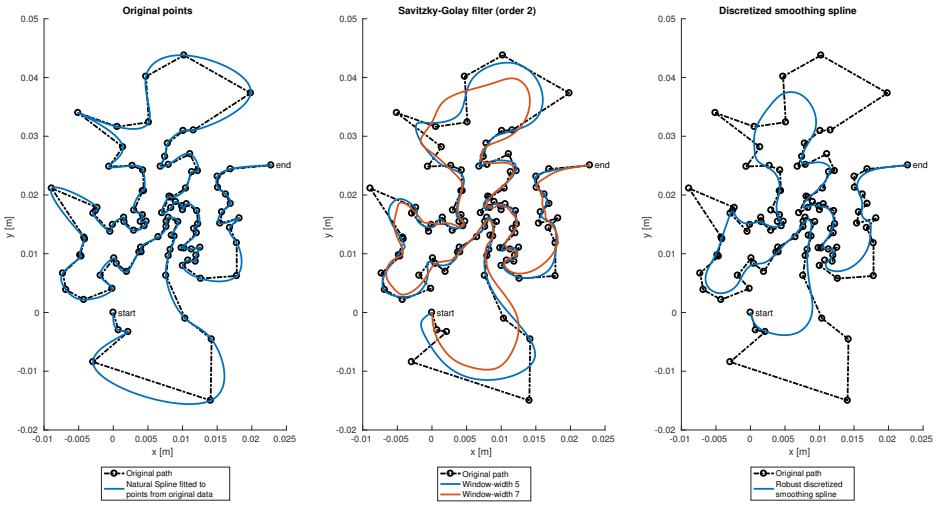



Figure 21: The different filtering and smoothing methods: The left figure shows a natural spline fitted to the optimal route. The middle figure shows the optimal path filtered by a Savitzky-Golay filter. The right figure shows the optimal path after being filtered by a discretized smoothing spline [43].

disadvantages that need to be discussed, in order to select the approach that is best suitable for this thesis. Figure 21 shows the results of the smoothing methods. The left frame shows the approach where a natural spline was fitted to the sampled points. As can be seen from this, the trajectory passes directly through each point of the originally calculated path. This means the trajectory is, even though continuous, producing abrupt changes in direction of motion (see the spike in the top left corner). Since the points were sampled randomly, it is not especially important to fully adhere to the sampled points. Thus two other methods are presented that filter the originally calculated path through the points. The first method is the use of a Savitzky-Golay filter, the result of which can be seen in the middle frame of Figure 21. In Chapter 3.3.2 it was mentioned that the Savitzky-Golay filter is especially useful when preservation of features is necessary. The figure shows that in so far that the general shape of the resulting trajectory is preserved, no matter which window-length of the filter is chosen. This is in stark contrast to the trajectory that results from the discretized smoothing spline (Figure 21, right frame). Here many of the outer parts of the trajectory are seemingly ignored. This is presumably because of the robustness against outliers that was discussed in Chapter 3.3.2.

Thus, ultimately the Savitzky-Golay filter was chosen to smooth the calculated optimal path. Regarding the window-length of the filter, a compromise between accuracy and smoothness is achieved: The increased window-length of 7 provides a smoother path through the densely sampled area than a window-length of 5, while not losing too much information of the outer areas of the trajectory. Thus for each trajectory generated in the concluding experiments (Section 5.5), a Savitzky-Golay filter with window length 7 was used for smoothing.

Another parameter that may influence the results is the choice of subsampling distance between each point of the smoothed trajectory. As explained in Section 3.3.2, the subsampling is needed to convert the continuous smoothing function of the Savitzky-Golay filter into discrete steps, which then are used for force prediction (see Chapter 3.2.3).
From Figure 22 can be seen that a subsampling distance of $5 \times 10^{-4}$ provided both a low enough set of sample points, as well as a smooth path. Thus every trajectory generated for the concluding experiments in Section 5.5 was subsampled with distance of $5 \times 10^{-4}$.

## 5.4 Choice of controller

In Chapter 3.4, the impedance controller with force-feed-forward was presented, as well as two force-only controllers as comparison. In the following sections, these controllers are evaluated. To do this, a few episodes with different configurations were generated to investigate the different behaviours of the robot. The following sections present the corresponding experiments and their results.

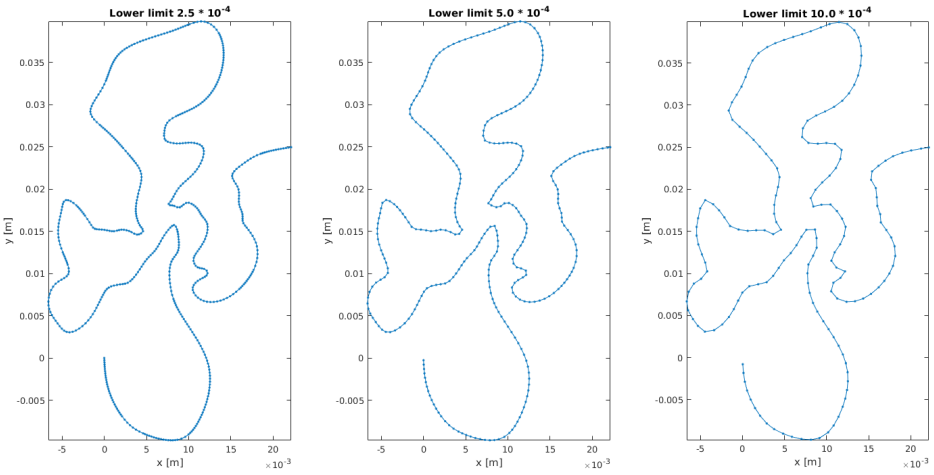



Figure 22: Three different distances between two successive points, subsampled from the filtered, continuous trajectory function. The trajectory shown is not used in this this, however it still demonstrates the effect of the different sampling-distances.

### 5.4.1 Impedance Controller

The two episodes supplied to the impedance controller consists of 100 and 300 sampled points respectively, which were both filtered by a Savitzky-Golay filter (order 2, window-length 7) and the corresponding predicted forces. The impedance controller was set to have a stiffness of $1000\frac{N}{m}$ and a damping coefficient of $60\frac{Ns}{m}$ and the forces are forwarded 1:1.
An example of these trajectories can be seen in Figure 23, where the reproduced trajectory can be seen in comparison with the unfiltered optimal route through the sample points and the smoothed one. If this figure is examined with respect to the desired and reproduced trajectories, one key element stands out: the reproduced trajectory looks like a strongly filtered version of the commanded smoothed trajectory.

### 5.4.2 Effect of predicted forces

Since one part of the thesis is to keep the framework as simple as possible, it is investigated if the feed-forward component of the impedance controller is required. For example, if the feed-forward forces produce no effect, the generation of forces can be removed from the framework. For this, first the accuracy of the predicted forces (see Chapter 3.2.3, Equations 7 - 12) is examined in order to gain insights into the generation of forces and thus to better understand the results of the experiments.

To do this, the originally demonstrated states $\Delta x$ and $\Delta y$ are given as input $\boldsymbol{x_{cond,2}}$ in Equation 10, in order to compare the predicted forces to the ones that

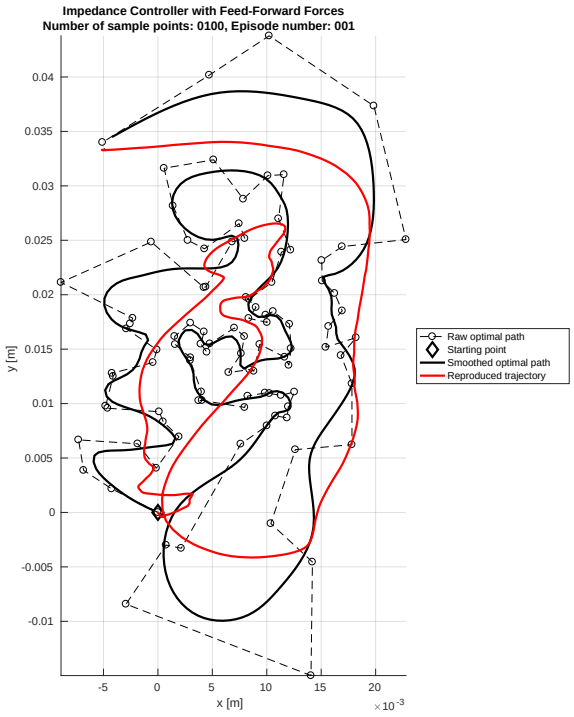



Figure 23: This figure shows the result of an episode with 100 sample points on the impedance controller with force-feed-forward, which ended unsuccessfully. The reproduced trajectory (red) differs from the smoothed trajectory (solid black).

were originally recorded. The resulting comparison is visualized in Figure 24, where the originally demonstrated forces in black and the generated forces in red are shown at each demonstrated position. The figure shows that the direction of the generated forces is predicted correctly most of the time, while the absolute value of the predicted forces is generally smaller than the original ones.

Before assumptions can be made it is however necessary to investigate the predicted forces of an actual learned trajectory to see the complete effect. Figure 25 thus presents the path and its predicted forces that result in the trajectory shown previously in Figure 23. By examining the figure closely, one can concede that many of the predicted forces are partly incorrect. Especially the x-component of the forces for those parts of the trajectory that move from positive to negative x-direction are smaller compared to those from negative to positive direction. Overall the forces' general order is correct, considering they are generated from only a single multivariate Gaussian. Since the distances between each point of the trajectory are smaller in the generated trajectory than in the demonstrated one, the encountered forces should be smaller as well. This is correctly predicted, as the figure shows smaller forces (arrows

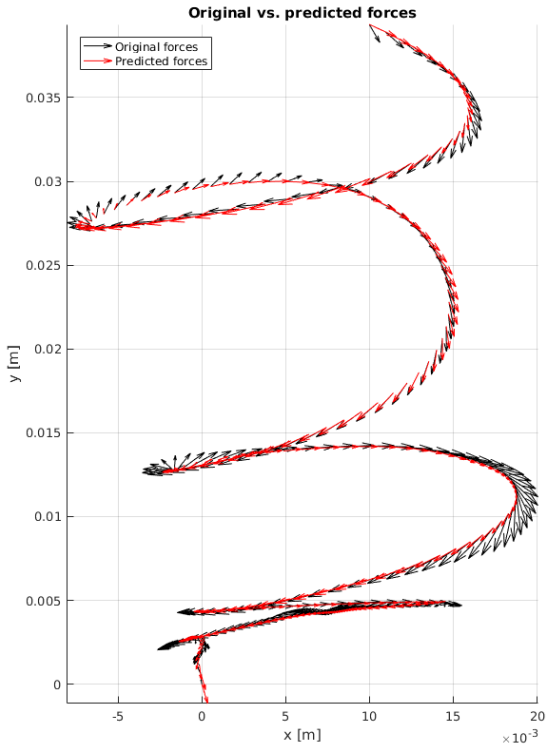



Figure 24: The accuracy of the predicted forces is tested on the data of the original demonstration. It can be seen that the predicted forces point roughly in the same direction as the originally recorded ones, although they tend to be much smaller.

in the figure) for the generated trajectory. Nevertheless, incorrectly predicted forces might affect the performance of the learning framework dramatically. Thus a first set of trials was performed which provide more information, before the effects are studied in the final experiments of Chapter 5.5.

In order to conduct the first trials, the force-feed-forward component was removed from the controller. Thus the only forces the impedance controller commands to the robot result from the difference between current and next state. Otherwise the settings are the same as in the previous trial. The episodes supplied to the impedance controller without feed-forward are the same as before, allowing a fair comparison between the two controller types.
The results show that the difference between the two recorded trajectories is not high (see Figures 26 and 27). It is observed from the two figures that the trajectory with the predicted forces differs only a maximum of ca. $3 - 4mm$ from the one without force-feed-forward. While these two figures are not significant on their own, it is nevertheless assumed they represent the complete behaviour. As such, the hypothesis is made that, due to the small differences in trajectory, the addition or removal of feed-forward forces does not have a big impact on the outcome of the project. However, as mentioned earlier, this is evaluated more thoroughly during the final experiments in Chapter 5.5.

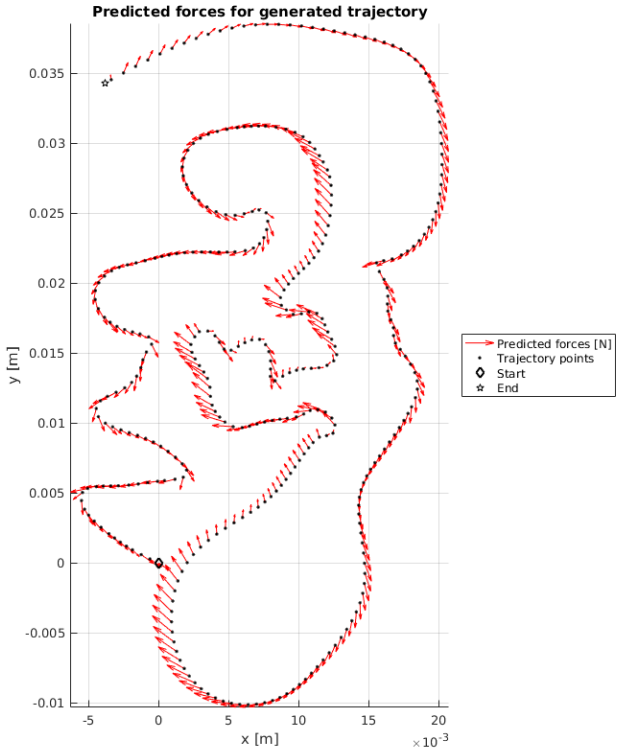



Figure 25: The smoothed path and its forces.

### 5.4.3 Force-only control

In order to investigate if an easier controller (see Chapter 3.4.2) allows for similar results as the impedance controller, a comparison is made between them. The force controller – be it with feedback loop or without – is a basic approach to reproduce the learned search. Since no positional data is required, only the predicted forces are used to move the robot. In the following, both versions of force control – with and without feedback – is studied.

First, two short trials were conducted for the force-feed-forward controller: one experiment where the predicted forces are played back unmodified and one where the predicted forces are multiplied by two. The results of the first trial show that with the generated forces alone, the robot did not move. Even with other sets of forces, the result is the same: the peg did not move. The behaviour persisted even when using the originally demonstrated forces. In order to check if the robot was stuck or not, the peg was manually pushed a few centimeters into several directions. Not even after these kind of disturbances did the peg move by itself. In the second trial, the forces were subsequently multiplied by a factor of two, five and ten. The first two factors (two and five) had no influence on the outcome of the reproduction, the peg did not move. However, once the forces were multiplied by ten, the peg started to move in exaggerated manner and finally slid off the platform. The result of this reproduction is visualized in Figure 28. The trajectory was manually stopped so

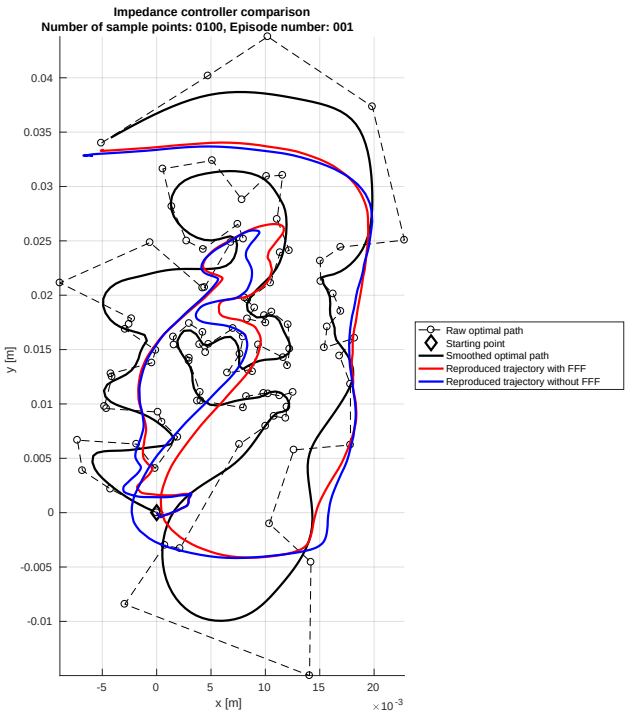



Figure 26: This figure shows the comparison of one episode (100 samples) executed by the impedance controller with force-feed-forward (red) and without it (blue). Both trajectories do not differ more than a few millimeters from one another.

that the robot would not harm itself or the environment. A detailed review of this behaviour is presented in the discussion part of this chapter (Section 5.4.4).

After this experiment, the force-feedback controller was evaluated. For this controller, instead of using a set of predicted forces, the originally demonstrated forces are used, so that only the performance of the controller is studied. To do this, the forces that were recorded during the demonstration are played back to the force controller. The proportional and integral gains are set to $K_p = 1$ and $K_i = 75$ respectively, which are the values that were experimentally found to perform best. The reason the proportional gain is so low is for gains above 2, the robot's motions becomes increasingly jittery without moving far, as seen in Figure 29. After the parameters are set, the trajectory is reproduced two times, one directly after the other. During both reproductions, all settings are kept the same. The resulting trajectories are seen in Figure 30. From this it is obvious that the first produced trajectory (seen in blue) is too small to be used. Although the second trajectory (seen in orange) is more suitable, it is still too small in y-dimension. Although the shape of the trajectory resembles that of the commanded positions (black dashed lines), the reproduction is too skewed to be considered a good imitation.

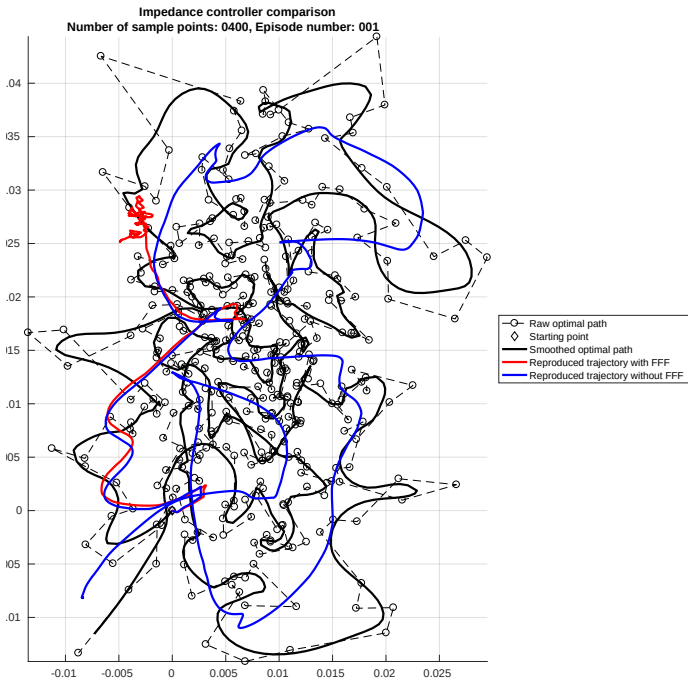



Figure 27: This figure shows the comparison of another episode (400 samples) executed by the impedance controller with force-feed-forward (red) and without it (blue). Even though the difference between the two trajectories is small, the controller with force-feed-forward reaches the goal, as opposed to the one without it.

These force controller trials indicate that it seems in fact impossible to learn and reproduce a search motion only by use of interaction-forces. Instead some kind of positional control is necessary, e.g. that of an impedance controller. The reasons for this is discussed in the second part of the following Discussion section.

### 5.4.4 Discussion

One observation made during the trials with the impedance controller is that, while the reproduced trajectory does indeed follow the generated trajectory as expected, the level of detail on the reproduced trajectory is much lower than that of the commanded one (see Figure 23 and 19). It looks like a strongly filtered version of the generated smoothed trajectory. While it is impossible to determine the cause of this accurately, it is likely that the behaviour is caused by either the friction forces between the peg and the surface of the cylinder, the internal joint frictions of the robot, or both. These friction forces presumably are on the same level as the ones from the positional

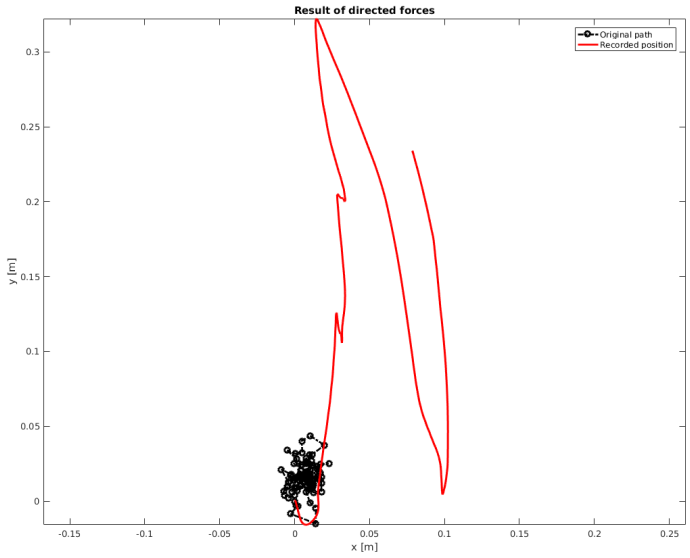



Figure 28: This figure shows the exaggerated trajectory the force-feed-forward controller executed if the supplied forces are multiplied by a factor of ten.

part of the impedance controller. This, in turn, means that there are two possibilities to solve the problem, both with their advantages and disadvantages.

The first possibility is to increase the stiffness parameter of the impedance controller to, *e.g.* $2000\frac{N}{m}$. This increases the forces the impedance controller returns for the positional difference and would allow for a more precise pursuit of the commanded trajectory. However, as explained in Chapter 3.4.1, increasing the stiffness also means that the motions are less compliant. Additionally, the velocities of the motions increases. These two effects might have a big impact on the success rate of the task, as less compliance and higher velocities mean that the peg needs to go precisely over the hole's center and not just the $4 - 5mm$ circle around it. Even then, if the velocity is high at this point, chances are the peg might pass over the hole. This is something to consider in future work.

The second method to assure more precise trajectory following is to allow for more time between each commanded trajectory point. By allowing more time to reach its goal, the trajectory can be followed more closely without sacrificing compliance. This however will presumably increase the time until completion, another issue discussed in the next paragraph. Additionally, during general examination of the robot and its motions, it was observed that low forces are sometimes not enough to overcome the combination of internal joint friction and surface friction. It thus stands to reason that this method might not be as accurate in tracking the commanded trajectory as increasing the stiffness is.

As mentioned in Section 5.4.3, all of the experiments done with pure force-control indicates that it is impossible to imitate search motions merely with the use of forces.

Due to the unexpected results, a thorough discussion is necessary to determine the reason behind them. The first experiment to discuss is that of the force-feed-forward controller. Since it is clear from the results that merely playing back the forces, even the ones originally recorded, is not enough to move the robot. The reasons for this behaviour stems from either the surface friction or internal joint frictions. The points that suggest the surface friction is the cause of the strange behaviour are: once the forces are big enough, in this case they needed to be multiplied by ten, they overcome the static friction between the peg and the surface. Because the static friction is usually higher than the dynamic friction [52], once it is overcome, the additional forces that were needed to get over this threshold are used to accelerate the peg in x- and y-direction. This could be the exaggerated motion seen in Figure 29 (red line), meaning that the forces originally recorded are only those required to overcome the dynamic friction. However, this stands in contrast to the observations made when manually pushing the peg during reproduction, because once the peg was pushed, the static friction was overcome. It thus stands to argue if it is the true reason behind the behaviour.

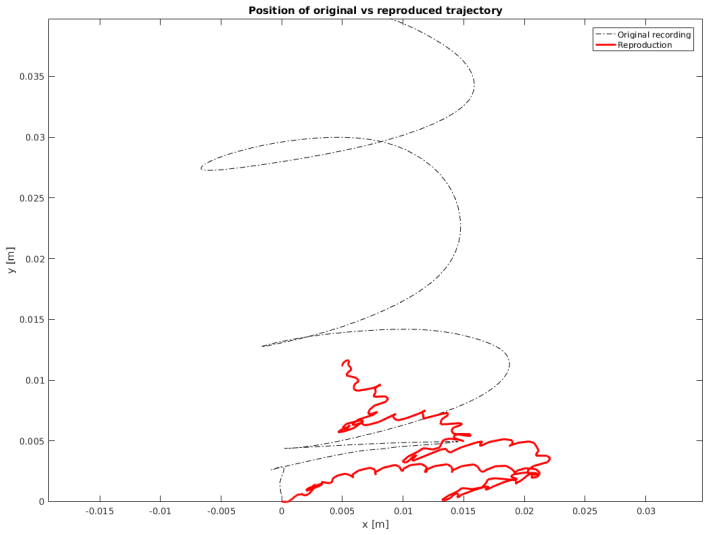



Figure 29: This figure shows the jittery trajectory a force-feedback controller with high proportional gain ($K_p = 13$) produces. These kind of motions are not desirable and should be avoided since they keep losing surface contact. Thus the proportional gain has to be kept low.

Another reason could be the internal joint frictions of the KUKA robotic arm. Principally, the underlying reason is the same. The static friction present in the seven joints of the robot is much higher than their dynamic friction. Thus, once this threshold is overcome, the force requirement to move the joints is greatly reduced. As opposed to the surface friction, the joint frictions are present in each of the robot's joints differently. Thus it is much more difficult to discern which joint – or joints –

cause these high friction thresholds. This could explain why manually pushing the peg did not help setting it in motion, while the ten-fold multiplied forces did: the pushing motion might not have moved the joint responsible for the high friction, while the commanded joint torques from the KUKA's internal controller did in fact move it enough. Additionally, the joint frictions were not included in the measured forces, since those are only include forces acting directly on the peg. However, without a model of the joint frictions it is difficult to make complete and certain assertions. Thus, in order to accept or reject the hypothesis that the joint frictions are responsible more data and information about the friction model of the robot is needed.

The experiments with the PI force-feedback controller pose even more unexpected results. While being slightly better than the feed-forward controller, the force-feedback results show it is unfeasible to reproduce a learned search only with the prediction of forces. Since the results are the product of the originally demonstrated forces, it is clear that the issue lies within the controller, not the prediction algorithm. Another unexpected result can be seen in Figure 30, which shows that two exactly identical configurations can lead to two different outcomes. The reason, however, is rather obvious: the bigger trajectory of the two (the orange line in Figure 30) is appearing *if and only if* the reproduction is started immediately, i.e. within half a minute, after the end of another reproduction. This seems to show quite clearly that the internal joint friction is the main issue of this behaviour. The previous reproduction has heated up the joints in such a manner that the friction coefficients seem to drop considerably, and continue to stay low for around 30 seconds. This enables the immediately following second reproduction move in the shown fashion.

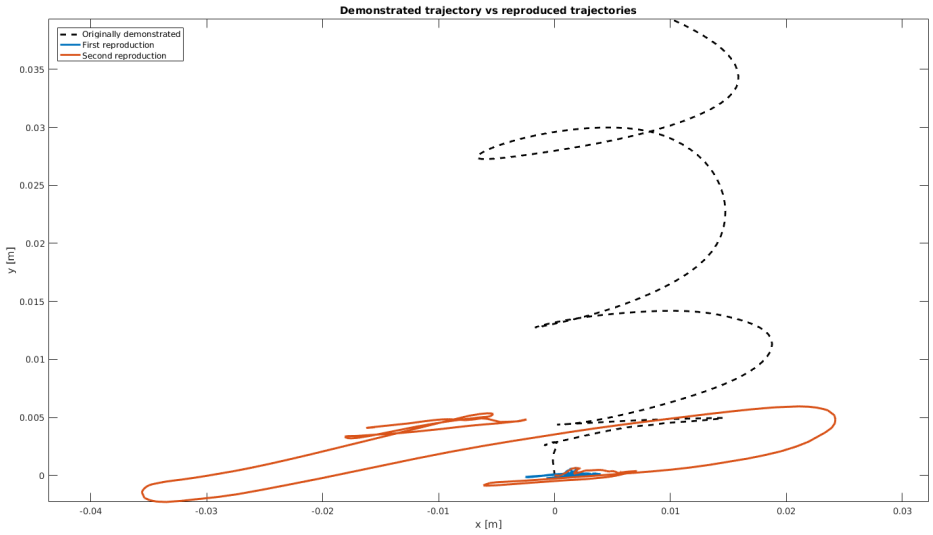



Figure 30: This figure shows two different reproductions of the same set of forces. The smaller, blue trajectory represents the first reproduction, and the orange trajectory is from the reproduction that was started immediately after the first one.

In order to investigate this issue further, the interaction forces during both reproductions were recorded and are compared to those that were recorded during the original demonstration. This comparison is seen in Figure 31 for the first trajectory, and in Figure 32 for the larger trajectory. The figures show that both reproductions are imitating the originally demonstrated imitation forces well, especially the earlier, smaller trajectory (Figure 31). If the two figures are compared, both forces are about the same, although the ones of the larger trajectory seem to be a bit high, but only by about 0.5 to 1 Newton. These additional Newtons might be the reason for the larger trajectory. Thus, it can be surmised that a higher temperature of the joints leads to reduced joint friction, which in turn allows for more accurate execution of forces. These forces then lead to a larger trajectory. This, however, does not fully explain why the distance moved in x-direction is much larger than that in y-direction. Nevertheless, it is clear that the friction of the internal joints has some effect on the reproduction of forces.

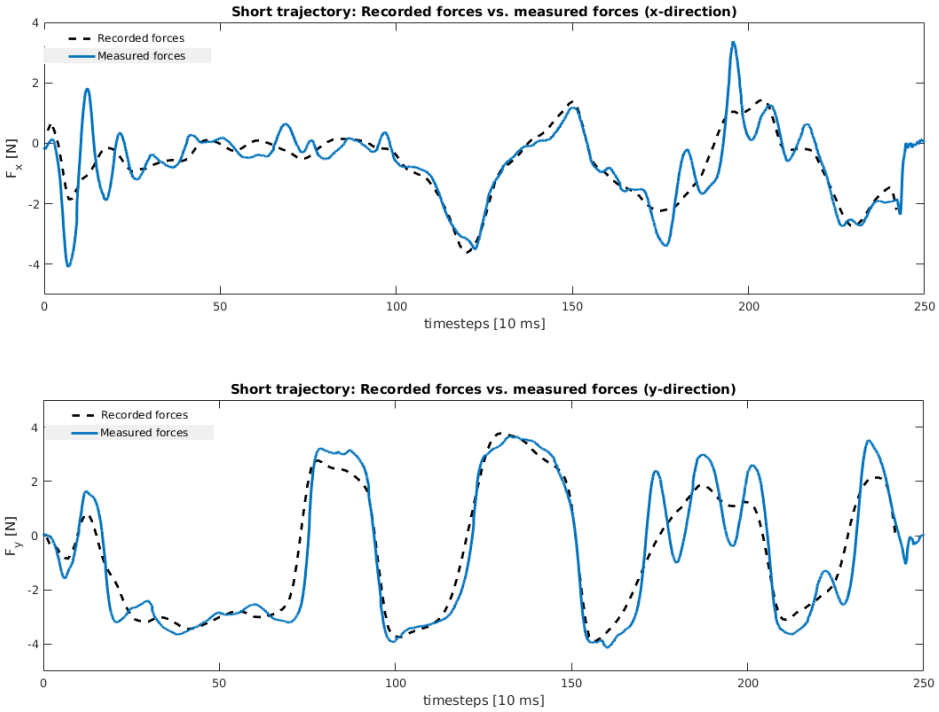



Figure 31: The figure is divided into two parts. The upper part shows the x-axis and the lower part the y-axis. The figure shows the comparison of the interaction forces measured during the first reproduction (solid blue line) with those of the original demonstration (dashed black line). It can clearly be seen that the reproduction follows the original forces closely. There are a few oscillations, which are caused by the high integral gain of the controller.

Beside this, Figure 31 shows an even more interesting behaviour: Although the originally demonstrated forces are reproduced almost identically, the resulting motion of the peg is no comparative to the demonstrated motion. As with the previous issues, this is presumably caused by either internal or surface friction. The exact reason is, just as before, impossible to deduct without a detailed model of the joint frictions and more data of the surface friction. All in all, these results present a significant observation which is: Without further additions to the controller that take care of overcoming static friction, it is impossible to imitate search motions for the peg-in-hole environment with force-control only. To pursue the idea of a purely force controlled search imitation, it is strictly necessary to provide the algorithm with a detector that recognized the threshold between static and dynamic friction. Therefore, robots that provide a model of their internal frictions would be more suitable for such approaches.

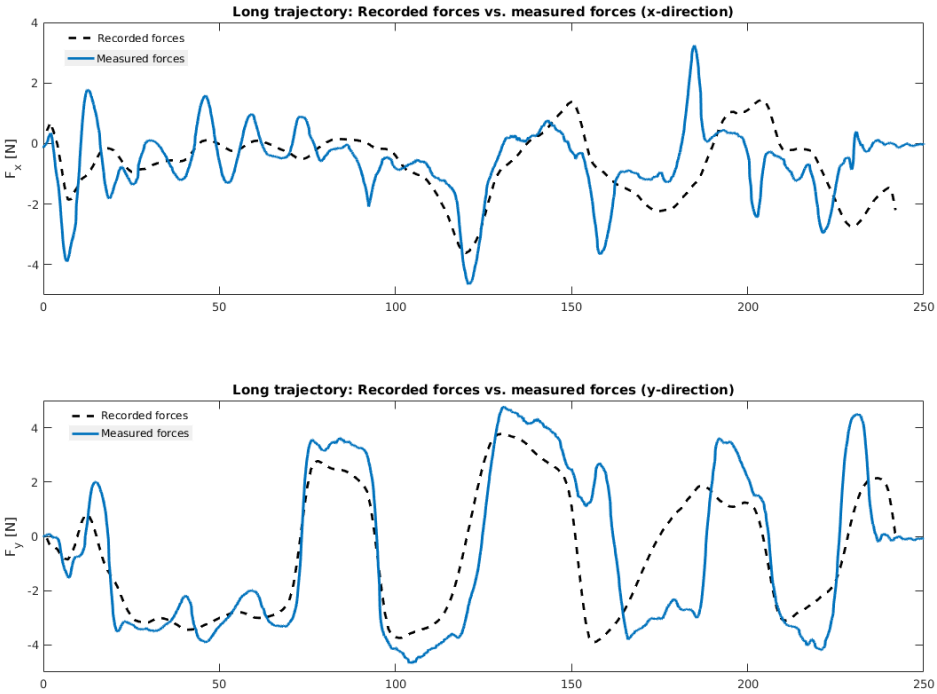



Figure 32: Similarly to Figure 31, this figure is split into two sections. The above section shows the x-axis while the lower section presents the y-axis. The comparison shows the measured interaction forces during the second reproduction and the ones from the original demonstration. The original forces are still reproduced well, however, not as ideally as those of the first reproduction.

## 5.5 Experimental evaluation

After the effects of the different parameters of the learning framework were evaluated and the first trials of the controllers were discussed, it is necessary to produce a more statistically significant experiment to fully validate the assumptions made during the previous sections. The findings of the previous sections are used to set up the experiments correctly. This allows for an outcome that is as clear and concise as possible.
According to Chapter 5.1, the original demonstration is regarded as a worst-case-demonstration and thus the starting position is set closer towards the hole to resemble a more average search. Chapter 5.2 shows that more sample points lead to a higher coverage and it was assumed that this leads to a more successful search. This is something that needs to be evaluated on a big scale in order to prove this assumption. In Section 5.3 the Savitzky-Golay filter (order 2, window size 7) is chosen as the best of the presented smoothing methods. Thus that method is used to smooth any generated trajectories. Additionally, this section also showed that a subsampling rate of $5 \times 10^{-4}$ allows for both preservation of the smooth trajectory as well as enough space between its individual points, and thus this rate is used to subsample the smoothed trajectories. Furthermore, it was hypothesized in section 5.4.2 that the feed-forward forces have little influence on the outcome. Thus two experiment series were conducted: one for the impedance controller with force-feed-forward and one for the impedance controller without the feed-forward component.

First, the experiments for the complete impedance controller with force-feed-forward were conducted. To supply a reasonable number of episodes – i.e. a generated trajectory plus its corresponding predicted forces – were generated in order to show statistically significant results. Thus, for a set of 100, 200, 300, 400, 500 and 600 sample points each, a series of 20 randomly generated episodes was created. This not only proves or disprove the assumptions made in Chapter 5.2 about the effects of the sample size, but also leads to more statistically significant results than the single trials made earlier.

In addition to this, a second series of experiments was conducted for the feed-forward-less impedance controller. Principally, the same settings were used as for the experiments for the feed-forward impedance controller, i.e. the starting position, filter and subsampling were identical to those used in the full impedance controller. However, due to time limitations the choice of sample sizes was reduced to sets of 100, 200 and 400. Nevertheless, for each of the sample sets, a series of 20 episodes was generated to allow for more statistically significant results.

The following section shows the results that were gained from the described experiments, which where all conducted with a stiffness of $1000\frac{N}{m}$, and a damping of $60\frac{Ns}{m}$.

### 5.5.1 Results

First, the results of the 120 episodes are presented that were gained from the impedance controller with force-feed-forward. As can be seen from the numbers in Table 1, these results are promising. At 300 samples, the success rate was 85%. This shows that the proposed framework is able to learn the demonstrated search for an assembly task with just one human demonstration.

| Sample size | 100 | 200 | 300 | 400 | 500 | 600 |
|---|---|---|---|---|---|---|
| Success rate | 6/20 | 12/20 | 17/20 | 17/20 | 17/20 | 14/20 |
| Success percentage | 30% | 60% | 85% | 85% | 85% | 70% |
| Incl. partial insertions | 50% | 75% | 90% | 90% | 95% | 85% |

Table 1: The results for the impedance controller with force-feed-forward show that 300 sample points is already a sufficient amount of coverage, if the starting position is chosen wisely. The fourth row shows the success percentage if partial insertions count as successes as well.

The increasing success rate from 100 samples to 300 also confirms the hypothesis made in Section 5.2, that the success of the search increases with higher sample sizes. However, after 300 samples the success rate consistently stays at 85%, no matter how much more the sample rate is increased, with the exception of 600 samples, where it declined to 70%. This behaviour stands in contrast to the assumption of Section 5.2 and thus needs to be discussed further in the following discussion section (see Chapter 5.5.2).

Additionally, during the majority of the of the unsuccessful episodes — 25 out of 37, 68% —, it was observed that the peg got close to the hole, but missed it by a few millimeters. Even more so, during 14 of the unsuccessful episodes, the tip of the peg actually slid a few millimeters into the hole, but then jumped out of the hole again. If those 14 episodes with partial insertion were added to the successful ones, the success rate increases for each sample size. Especially lower numbers seem to benefit from it, as can be seen in Table 1 in the fourth row. In Section 5.5.2 possible improvements for these partial insertions are discussed.

These results also answer the question as to what kind of influence the randomness of the generated trajectories has, or rather it poses a requirement to reduce the influence of said randomness: Only if many episodes are generated, a bigger picture of the actual chance of success is gained.

Since Chapter 5.4.2 shows that the trajectories without feed-forward are not much different than those of the impedance controller with force-feed-forward, which is why the assumption was made that the predicted forces have no outcome on the final results. However the evaluation of the experiment series for the feed-forward-less

impedance controller prove this assumption to be wrong.
The results of these experiments, seen in Table 2, show that that the rate of success is consistently lower than that of the force-feed-forward impedance controller, even if partial insertions are included. The success rate of the feed-forward-less method lies consistently at least 25% beneath that of the impedance controller with feed-forward forces. Since these results stand in such contrast to the assumption made in Section 5.4.2, they are discussed in Section 5.5.2.

| Sample size | 100 | 200 | 400 |
|---|---|---|---|
| Success rate | 2/20 | 7/20 | 13/20 |
| Success percentage | 10% | 35% | 65% |
| Incl. partial insertions | 10% | 35% | 70% |

Table 2: The success rate of the impedance controller without force-feed-forward still increases with higher sample size, but nevertheless is smaller than that of the controller with feed-forward forces.

### 5.5.2 Discussion

The first experiment series showed that is in fact possible to learn the search motions necessary for a successful reproduction from a single human demonstration. This was indeed expected after it was shown in Sections 5.1, 5.2 and 5.3 that the generated trajectories can cover the goal area under the right circumstances. However, the high number of successful imitations comes in fact a little unexpected. This high number might indicate that the number of sample points per episode is set high compared to the search space and thus the area is already covered enough at 300 samples.
That also falls in line with the other observation made during that experiment: The success rate rises from 30% at 100 samples to 85% at 300 samples and subsequently stagnates with higher sampling rates (see Table 1), even though more samples should mean more coverage and thus a higher chance of finding the goal. One reason for this might be that due to the location of the hole towards the low-density area of the Gaussian distribution, the chance of sampling points there does not increase much. The other unexpected observation is that at 600 samples, the success rate suddenly decreases by 15%. This might however merely be a statistical outlier. And even though all but one of the unsuccessful episodes were at some point of the reproduction observed to be close to the hole, this applies for the series with lower sample sizes as well. However, without further data this is the only feasible explanation at this state, and barely more than speculation. Doing more experiments with an even higher number of sample points would be a good way to investigate this behaviour in a possible future expansion of this work.

The second experiment's results presented in Section 5.4.2 paint a unexpected but nevertheless useful picture. Even though the difference between the two controllers is not much, the impedance controller with force-feed-forward succeeds much more

often than the one without the predicted forces: As mentioned earlier, the success rate of the feed-forward-less impedance controller is consistently $25 - 30\%$ lower than that of the force-feed-forward impedance controller. That means that the usage of predicted forces consistently improves about four to five episodes to succeed. Since there are only 20 episodes per sample size, it is possible that this is merely a product of an unlucky draw of episodes. Another possible reason for this behaviour is of course that the learning framework did indeed predict the necessary forces correctly, i.e. learned them correctly from the human demonstration. This would not only show that it is indeed necessary to learn the forces from human demonstration, it would also justify using the impedance controller with force-feed-forward. Because of the consistent improvement with the use of predicted forces, this seems a more likely explanation than the forces merely being a lucky draw.

While there is no way to deterministically identify the true reason behind this behaviour, it is however possible to make a statement about the statistical significance of the two results. Thus, in order to test the independence of the two results, Fisher's exact test [53] is used. Table 3 shows the data used for the test. While this table only shows the results from the set of 400 sample points, the test was also performed for the sets of 100 and 200 sample points. For 400 samples, the result of Fisher's exact test shows the hypothesis that the higher success rate of the impedance controller with force-feed-forward is the product of randomness can only be rejected with $p = 0.104$. Similar values result from the other sample sizes: for 100 samples $p = 0.0958$ and for 200 samples $p = 0.0744$. As the usual accepted level for statistical significance lies at $p \leq 0.05$, the results presented in this thesis cannot be regarded as statistically significant. However, since the results are close to that, it is recommended to conduct the experiments with more episodes in the future.

| | With feed-forward | Without feed-forward | Totals |
|---|---|---|---|
| Successes | 17 | 13 | 30 |
| Failures | 3 | 7 | 10 |
| Totals | 20 | 20 | 40 |

Table 3: Fisher's exact test is used to inspect if the higher success rates of the impedance controller with feed-forward are statistically significant. The results of the test show the notion that the higher success rate of the impedance controller is the product of randomness can only be rejected with a probability of $p = 0.104$.

### 5.5.3 Limitations

There are a few limitations that are present throughout all of the experiments mentioned above. One such limitation that affects all of the experiments done in this thesis is the fact that only 20 episodes were generated for each series of experiments. While this does in fact give a good idea of the overall behaviour of the algorithm, it is nevertheless difficult to make assertions about the behaviour at all times. Generating

more episodes, *e.g.* 100 episodes per series, would be a good way to make sure that the experiments at hand are not merely the result of a lucky or unlucky series of random samplings. This might also give more insight into the problem encountered earlier with the decreased success rate at 600 samples. While this is a rather laborious task it is nevertheless a helpful improvement for further continuation of this work.

Another limitation of the experiments done in this thesis is linked to the issue of time. While the time until insertion can be gathered from the recorded data, it was out of the scope of this thesis to include it as a factor in the evaluation. It is thus impossible compare the completion time of the generated trajectories with that of the original human demonstration. However this is a interesting and important aspect that can play a big role in deciding if the imitated searches are on the same level as those of a human. Even more so, it would help as well with respect to deterministic search strategies that were presented in Chapter 2.4, especially the Archimedean spiral pattern proposed by [21]. This might even be the subject of a new study, where the average completion times, as well as the worst- and best-case times are compared between the deterministic and the human learned search strategies are compared.

Yet one more limitation of the presented results is that they only show that the algorithm works for a 2D environment. The proposed algorithm however is designed so that it works for 6D and even n-dimensional searches as well. In actuality, a six-dimensional use case was originally planned to be part of the thesis but then was scrapped as it would go beyond the scope of this thesis. This would be a good point to start if the work is to be continued in the future.

Another finding in the results was that in many of the unsuccessful episodes, there were partial insertions, i.e. moments where the peg did slide a few millimeters into the hole, but then moved upwards out of it again. This was rather unexpected, but nevertheless comprehensible if the cause behind those instances is clarified. The reason why the peg jumped out of the hole was because the level of downwards directed force was not enough as compared to the forces from the impedance controller moving the peg further away from the hole. This in turn means that either a downward force value or a lower stiffness parameter might lead to these partial insertions to be completed when they occur. However, both of the solutions have a deep impact on the dynamics and might not end up helping the success rate increase after all. This is because a higher downward force also means more friction between the peg and the surface. This could mean that the previous stiffness of $1000\frac{N}{m}$ is not enough anymore to move the robot, and thus the actual coverage area might be smaller and not reach the goal area anymore. Similarly, decreasing the stiffness makes the motions more compliant, but also increases the ratio of friction vs. forces commanded by the impedance controller, which signifies a similar result as increasing the downward force. Investigating this relationship between vertical forces, friction and stiffness can easily fit into a study of its own, and would be helpful for any future work in this direction.

# 6 Conclusion

The goal of this thesis was to develop an algorithm that learns search motions from a blind human demonstration. The algorithm incorporates both the forces and the motions a blindfolded human uses during a search and imitate them in a similar fashion. In order to to this, a learning framework and corresponding controllers were developed in Chapter 3 and experimentally evaluated in Chapter 5.

The learning framework consists of a Gaussian distribution that models the search space and can predict forces, as well as a component that generates and filters a search trajectory. To provide the learning framework with the necessary data, a single blindfolded demonstration was made. In the learning framework, a single Gaussian distribution is fitted to the supplied data. Subsequently, a search trajectory is then created by sampling positions from the Gaussian distribution. The shortest possible route through these points is then calculated and the resulting path is filtered and smoothed by a Savitzky-Golay filter, as well as subsampled into a discrete set of coordinates. These coordinates are then used to predict the corresponding forces that will be encountered by the environment. This was done 20 times each for a series of 100 to 600 sample points per trajectory. Overall 120 different episodes were generated. To execute these episodes on a KUKA LWR4+ robotic arm, an impedance controller with force-feed-forward was developed. Additionally, two force-only controllers were developed as a comparison. Each of the generated episodes was then reproduced using the impedance controller. Additional experiments were done on the impedance controller without its feed-forward component, as well as on the force-only controllers. The experiments conducted in Chapter 5 showed that the method presented in this thesis can successfully learn search motions from human demonstration and reproduce them so that the search succeeds in most cases. They also showed that only the combination of impedance control and feed-forward of the predicted forces leads to a successful reproduction of the learned search, as both pure impedance control as well as force-only control showed inferior results.

However, there is a series of limitations that goes along with the discovered findings and claims made. On one hand there is the fact that the starting position of the search reproductions was assumed as an average position w.r.t. the demonstrated starting position, which was considered a worst case scenario. That means in order for the search imitation to be successful the starting position between demonstration and reproduction needs to be moved closer towards the goal. This was necessary due to the fact that the search space is modelled as a Gaussian. In the future, this problem could be solved by either using a better suitable or custom made type of distribution. This way, the probability density of the distribution could be weighed more towards the end of the trajectory, i.e. the goal of the task.
Additionally, the experiments were only conducted in a 2D environment, even though the algorithm is proposed to work in any dimensionality that can be demonstrated. This was due to the fact that additional multidimensional experiments were out of scope for this thesis. However, in the future task that include searches in both

position and orientation, such as the hose-coupling task presented in [16] can be used to validate the algorithm.
Another limitation of the algorithm in its current form is that it recognizes neither when a search begins, nor when the search is successful. Instead it relies on the compliance of the impedance controller to achieve completion. This is again due to the fact that developing a detection algorithm was out of scope for this thesis. However, the approach presented in [29] could be used in the future to detect when a normal task-operation becomes stuck, as well as to detect the transition from search back to normal task-operation.
The implementation of benchmark features such as completion time, as well as other search trajectories for comparison was outside the scope of the thesis, which presents yet another limitation. Due to this it is difficult to compare the presented search algorithm with the ones presented in Chapter 2.4. Thus in a future iteration of this work, deterministic search strategies such as the one presented in [21] and other interesting methods such as ergodic trajectories [26] could be implemented and compared with the approach in this thesis. This would greatly improve the measure of viability of the presented algorithm.

In conclusion, the algorithm presented in this thesis is a promising method to learn search strategies from human demonstration and merits further validation and development in multidimensional use-cases.

# References

[1] Kormushev, P., Calinon, S., Caldwell, D. G. (2011) Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input, Advanced Robotics, Volume 25 Issue 1, 581-603

[2] Argall, B. D., Chernova, S., Veloso, M, Browning, B. (2008) A survey of robot learning from demonstration, Robotics and Autonomous Systems, Volume 57 Issue 5, 469-483

[3] Nehaniv, C. L., Dautenhahn, K. The correspondence problem, in: Imitation in animals and artifacts, pp. 41-61, MIT Press Cambridge, MA, USA, 2002.

[4] Abbeel, P., Coates, A, Ng, A. Y. (2010) Autonomous Helicopter Aerobatics through Apprenticeship Learning, International Journal of Robotics Research, Volume 29 Issue 13, 1608-1639

[5] Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., Hebert, M. (2012) Learning Monocular Reactive UAV Control in Cluttered Natural Environments, International Conference on Robotics and Automation 2013

[6] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K. (2016) End to End Learning for Self-Driving Cars, Computing Research Repository, 1604.07316

[7] Demiris, J., Hayes, G. M. Imitation as a dual-route process featuring predictive and learning components: a biologically plausible computational model, in: Imitation in animals and artifacts, pp.327-361, MIT Press Cambridge, MS, USA, 2002.

[8] Ogino, M., Toichi, H., Yoshikawa, Y., Asada, M. (2006) Interaction rule learning with a human partner based on an imitation faculty with a simple visuo-motor mapping, Robotics and Autonomous Systems, Volume 54 Issue 5, 414-418

[9] Billard, A., Calinon, S, Dillman, R., Schaal, S. Robot Programming by Demonstration, in: Springer Handbook of Robotics, pp. 1371-1394, Springer-Verlag Berlin Heidelberg, 2008.

[10] Cohn, D. A., Ghahramani, Z., Jordan, M. I. (1996) Active learning with statistical models, Journal of Artificial Intelligence Research, Volume 4 Issue 1, 129-145.

[11] Schwarz, G. Estimating the dimension of a model, in: The Annals of Statistics, pp.461-464, Volume 6 Issue 4, 1978.

[12] Wasserman, L. (2000) Bayesian Model Selection and Model Averaging, Journal of Mathematical Psychology 44, 92-107

[13] Claeskens, G., Hjort, N. L. Model Selection and Model Averaging, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, UK, 2008.

[14] Calinon, S., Guenter, F., Billard, A. (2007) On Learning, Representing, and Generalizing a Task in a Humanoid Robot, IEEE Transactions on Systems, Man, and Cybernetics, Part B, Volume 37 Issue 2, 286-298

[15] Suomalainen, M., Kyrki, V. (2017) A geometric approach for learning compliant motions from demonstration, IEEE-RAS International Conference on Humanoid Robotics

[16] Suomalainen, M., Kyrki, V. (2018) Learning 6-D compliant motion primitives from demonstration

[17] Kronander, K. J. A. (2015) Control and Learning of Compliant Manipulation Skills, EPFL (doctoral dissertation)

[18] de Chambrier, G. P. L. (2016) Learning Search Strategies from Human Demonstrations, EPFL (doctoral dissertation)

[19] de Chambrier, G., Billard, A. (2014) Learning search policies from humans in a partially observable context, Robotics and Biomimetics, Volume 1 Issue 8

[20] de Chambrier, G., Billard, A. (2017) Fitted Policy Iteration for a POMDP Peg-In-Hole search task

[21] Jasim, I. F., Plapper, P. W., Voos, H. (2014) Position Identification in Force-Guided Robotic Peg-in-Hole Assembly Tasks, Procedia CIRP 23, 217-222.

[22] Jasim Ghalyan, I. F., Force-Controlled Robotic Assembly Processes of Rigid and Flexible Objects - Methodologies and Applications, Springer International Publishing, Switzerland, 2016.

[23] Chhatpar, S. R., Branicky M. S. (2001) Search strategies for peg-in-hole assemblies with position uncertainty, Proceedings 2001 Intelligent Robots and Systems, 1465-1470

[24] Abu-Dakka, F. J., Nemec, B., Kramberger, A., Buch, A. G., Krüger, N., Ude, A. (2014) Solving peg-in-hole tasks by human demonstration and exception strategies, Industrial Robot: An International Journal, Volume 41 Issue 6, 575-584.

[25] Mavrommati, A., Tzorakoleftherakis, E., Abraham, I., Murphey, T. D. (2017) Real-Time Area Coverage and Target Localization Using Receding-Horizon Ergodic Exploration, Transactions on Robotics, Volume 34 Issue 1, 62-80.

[26] Miller, L. M., Murphey, T. D. (2013) Trajectory optimization for continuous ergodic exploration, American Control Conference, 4196-4201

[27] Thomas, G., Chien, M., Tamar, A., Ojea, J. A., Abbeel, P. (2018) Learning Robotic Assembly from CAD, International Conference on Robotics and Automation

[28] Wrede, S., Emmerich C., Grünberg, R., Nordmann, A., Swadzba, A., Steil, J. (2013) A User Study on Kinesthetic Teaching of Redundant Robots in Task and Configuration Space, Journal of Human-Robot Interaction, Volume 2 Issue 1, 56-81

[29] Hagos, T. M., Suomalainen, M., Kyrki, V. (2018) Segmenting and Sequencing of Compliant Motions, Intelligent Robots and Systems

[30] Akgun, B., Subramanian, K., Thomaz, A. (2012) Novel Interaction Strategies for Learning from Teleoperation, AAAI Fall Symposium Series.

[31] Dietterich, T. (1995) Overfitting and undercomputing in machine learning. ACM Computing Surveys, Volume 27 Issue 3, 326-327.

[32] Bruyninckx, H., Dutré, S., De Schutter, J. (1995) Peg-on-Hole: A Model Based Solution to Peg and Hole Alignment, International Conference on Robotics and Automation, 1919-1924.

[33] Dillmann, R., Kaiser, M., Ude, A. (1995) Acquisition of Elementary Robot Skills from Human Demonstration, International Symposium on Intelligent Robotic Systems.

[34] Lin, L. L., Yang, Y., Song Y. T., Nemec, B., Ude, A., Rytz, J. A., Buch, A. G., Krüger, N., Savarimuthu, T. R. (2014) Peg-in-Hole assembly under uncertain pose estimation, World Congress on Intelligent Control and Automation.

[35] Morel, G., Malis, E., Boudet, S. (1998) Impedance based combination of visual and force control, International Conference on Robotics and Automation, 1743-1748.

[36] Nuttin, M., Van Brussel, H. (1997) Learning the peg-into-hole assembly operation with a connectionist reinforcement technique, in: Computers in Industry, Volume 33 Issue 1, 101-109.

[37] Yamashita, T., Godler, I., Takahashi, Y., Wada, K., Katoh, R. (1991) Peg-and-hole task by robot with force sensor: Simulation and experiment, International Conference on Industrial Electronics, Control and Instrumentation, vol.2, 980-958.

[38] Sena, A., Zhao, Y., Howard, M. J. (2018) Teaching Human Teachers to Teach Robot Learners, International Conference on Robotics and Automation.

[39] Eaton, M. L. (1983) Multivariate Statistics: a Vector Space Approach, John Wiley and Sons, 116-117.

[40] Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T. (1989) Numercial Recipes in C: The Art of Scientific Computing, Cambridge University Press.

[41] Vallabha, G. (2016) PLOT_GAUSSIAN_ELLIPSOID, in: MathWorks File Exchange, https://se.mathworks.com/matlabcentral/fileexchange/16543-plot_gaussian_ellipsoid (last viewed: 2018-07-30).

[42] Kirk, J. (2014) Traveling Salesman Problem - Genetic Algorithm, in: MathWorks File Exchange, https://se.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm (last viewed: 2018-07-23).

[43] Garcia, D. (2017) Robust spline smoothing for 1-D to N-D data, in: MathWorks File Exchange, https://se.mathworks.com/matlabcentral/fileexchange/25634-robust-spline-smoothing-for-1-d-to-n-d-data (last viewed: 2018-07-23).

[44] Garcia, D. (2010) Robust smoothing of gridded data in one and higher dimensions with missing value, Computational Statistics & Data Analysis, 54, 1167-1178.

[45] Steinmetz, F. (2014) Programming by Demonstration for in-contact tasks using Dynamic Movement Primitives, Master's thesis, Aalto University, School of Electrical Engineering.

[46] Tykal, M. (2015) Optimizing Programming by Demonstration for in-contact task models by Incremental Learning, Master's thesis, Aalto University, School of Electrical Engineering.

[47] Sharma, S. (2017) Control of Contact Forces between Robot and Free-Floating Object, Master's thesis, Aalto University, School of Electrical Engineering.

[48] Bischoff, R., Kurth, J., Schreiber, G., Koeppe, R., Albu-Schaeffer, A., Beyer, A., Eiberger, O., Haddadin, S., Stemmer, A., Grunwald, G., Hirzinger, G. (2010) The KUKA-DLR Lightweight Robot arm – a new reference platform for robotics research and manufacturing, in ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics).

[49] Schreiber, G., Stemmer, A., Bischoff, R. (2010) The Fast Research Interface for the KUKA Lightweight Robot, in IEEE ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications - How to Modify and Enhance Commercial Controllers.

[50] ATI (2015) ATI Force / Torque Sensor Mini45, https://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini45 (last viewed: 2018-07-24).

[51] Zhu, Z., Hu, H. (2018) Robot Learning from Demonstration in Robotic Assembly: A Survey

[52] van Beek, A. Coefficient of friction for a range of material combinations, www.tribology-abc.com, http://www.tribology-abc.com/abc/cof.htm (last viewed: 2018-08-01).

[53] Fisher, R. A. (1922) On the interpretation of $\chi^2$ from contingency tables, and the calculation of P", Journal of the Royal Statistical Society, Volume 85 Issue 1, 87–94